(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
4 January 2001 (04.01.2001)

PCT

(10) International Publication Number
WO 01/01300 A1

(51) International Patent Classification[7]: G06F 17/60

(21) International Application Number: PCT/AU00/00730

(22) International Filing Date: 28 June 2000 (28.06.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
PQ 1235 28 June 1999 (28.06.1999) AU

(71) Applicant *(for all designated States except US)*: INDUSTRY WIDE NETWORKS PTY LTD [AU/AU]; Level 1, 115 Clarence Street, Sydney, NSW 2000 (AU).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: HILSON, Daniel, Andrew [AU/AU]; Unit 1, 87 Macpearson Street, Waverley, NSW 2024 (AU).

(74) Agent: FREEHILLS CARTER SMITH & BEADLE; MLC Centre, Martin Place, Sydney, NSW 2000 (AU).

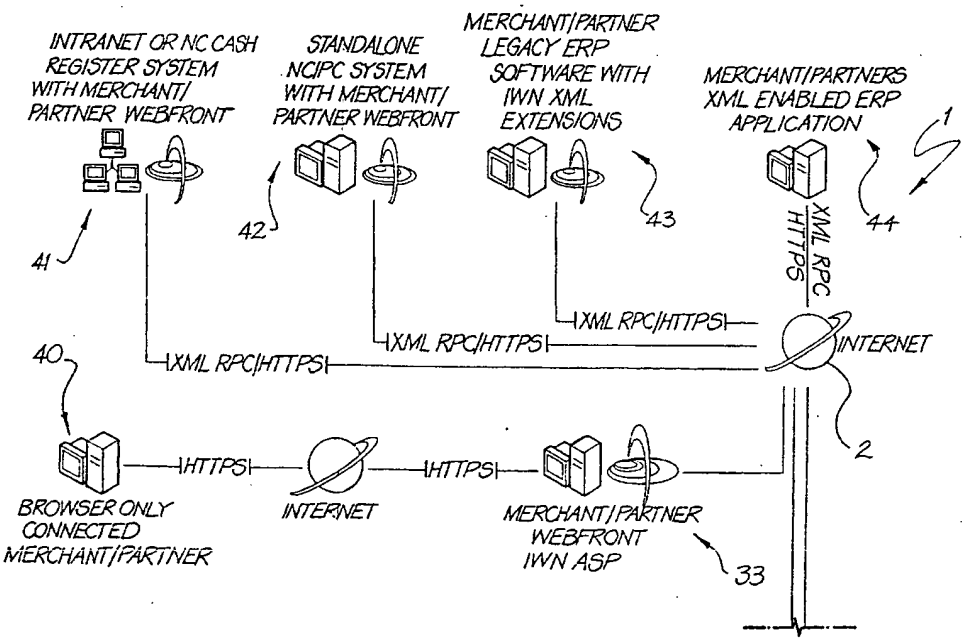
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *With international search report.*
— *Before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: AN INTERNET E-COMMERCE SYSTEM



(57) Abstract: An electronic commerce system comprising: a series of point of sale terminals providing for point of sale information handling of a business; an interconnection network interconnecting the point of sale terminals to a central database facility; a central database facility for storing information about each of the businesses for access by the operators of the point of sale terminals; and a series of service providers interconnected to the central database facility for meeting requests issued by the point of sale terminals.

An Internet E-Commerce System

Field of the invention

The present invention relates to the field of Internet electronic commerce and, in particular, discloses a hybrid e-commerce system having increased levels of functionality and operability.

Background of the invention

Traditionally, businesses have carried out activities utilising many different forms of communication. For example, letters, faxes and, more recently, e-mail have been traditionally used to place orders for the goods and services of a business.

Recently, the Internet has been undergoing an explosive growth period. In particular, the "World Wide Web" has provided a new avenue for the conduct of commercial transactions. This has lead to the concept of "E-commerce" in that commercial transactions can be carried over the Internet so as to facilitate a more optimal form of business operation. In particular it allows the direct selling of goods over the Internet from the producer to the consumer.

However, the utilisation of the World Wide Web often requires a high level of skill in the creation of HTML pages, Java scripts etc., in order to create appealing and attractive web pages having a high level of functionality. Additionally, a large number of different programs and hardware are often required. For example, browsers, e-mail programs, fax machine or fax software, a HTML editor, an ftp program etc.

Further, commercial business arrangements also often require separate Electronic Funds Transfer Point of Sale facilities for the conduct of sales transactions which are often totally separate from any Internet services.

As a result businesses, in particular, small business, tend to have a reduced uptake of Internet type operations.

Summary of the invention

It is the object of the present invention to provide for an E-commerce system having advantageous attributes.

In accordance with a first aspect of the present invention, there is provided an electronic commerce system comprising: a series of point of sale terminals providing for point of sale information handling of a business; an interconnection network interconnecting the point of sale terminals to a central database facility; a central database facility for storing information about each of the businesses for access by the operators of the point of sale terminals; and a series of service providers interconnected to the central database facility for meeting requests issued by the point of sale terminals.

Preferably, the system also comprises a series of suppliers interconnected to the central database facility for meeting requests issued by the point of sale terminals. The suppliers can include at least one of: an import/export agent; a warehousing agent or a producer. The service providers can include one of: a third party information vendor providing information upon request; a financial transaction vendor providing financial transaction authorisation upon request; or an order fulfilment vendor providing order fulfilment upon request. The point of sale terminals can include local database information and programs which are preferably downloaded on demand from the central database facility.

Access means for accessing the datastore as a member of the general public using a web browser and further means for communicating in a timely manner directly to the Point of Sale merchant and if that merchant can be there then communicating with the merchant in actual time.

The requests are preferably tranmitted in the form of XML documents or the like to and from the central database facility. The request implementation structure can be preferably provided by a software development kit applications programming interface.

The system can also include a series of user mobile data entry devices which interact with the point of sale terminals in the authorization of a transaction. The mobile data entry device can include one of WAP enabled phones, mobile phones or bluetooth connected devices.

Ideally, there is also provided a separate interaction unit such as a Web Browser for users to interact with the central database facility for the viewing of transaction statistics associated with the system. The viewing of transaction statistics preferably can include utilising OLAP facilities on the central database.

Actions undertaken by the database facility are preferably in the form of workflow steps executed by the facility with the workflow can be spawned by the template structure of the request. There can also be provided an interactive graphical database for interacting with the central database facility.

In further modified embodiments, multiple centralised database facilities are preferably peered and interact with one another to perform functions.

Brief description of the drawings

Notwithstanding any other forms which may fall within the scope of the present invention, preferred forms of the invention will now be described by way of example only, with reference to the accompanying drawings in which:

Fig. 1 illustrates schematically the arrangement of a first embodiment;

Fig. 2 illustrates the production of workflow in accordance with the first embodiment;

Fig. 3 illustrates the process of execution of transactions by a merchant/consumer and IWN engine arrangement;

Fig. 4 illustrates an alternative arrangment of an embodiment of the invention;

Fig. 5 illustrates the various structures in an embodiment of the invention

Fig. 6 illustrates the carying out of transactions in accordance with an embodiment of the invention; and

Fig. 7 illustrates one software architectural structure of the IWN engine.

Detailed description of the embodiments

In the first embodiment, an interactive system is provided for carrying out business transactions over the Internet in a simplified and automatic manner. The transactions can include electronic ordering and distribution, web site generation and on going maintenance, proactive interaction such as faxing, e-mail and electronic funds transfer. Further, the system allows business clients to maintain their own database and allows dynamic access to accounting procedures and management information systems including EFTPOS transactions.

Turn to Fig. 1, there is illustrated schematically the overall operational environment of a first embodiment. The first embodiment 1 is denoted the IWN network and includes various entities interconnected over an Internet environment 2. The elements communicate with one another using XML messaging and CORBA (common object request broker). CORBA is a general purpose communication protocol that allows transparent communications over a network. That means that the network is invisible to the development programmer in that they program as if the network is not there.

The object oriented communications model is used for application interfaces within IWN engine private processes. For example, the IWN server clearinghouse 30 requests transaction authorisation from a Payment Engine 36 via a CORBA RPC (remote procedure call). The architecture allows distribution of objects over the network. Objects can be written in one language (say Java) and referenced using another (say c++).

A series of java applets and servlets on the web server provide references to objects on other servers through the CORBA communication interface. The local client entities e.g. 40 can be configured to run Java applets over an Internet browser type environment such as that provided by Microsoft's Internet Explorer or Netscape's Netscape Navigator. Alternatively, they may be a permanent Java application, integration into a third party point of sale or accounting package, or a currently hardware of software device.

Each of a series of client side elements 41 - 44, 40, 34, 51, 52 interact with an IWN engine clearinghouse 30 which operates as a clearinghouse for operations. The flexibility of the arrangement allows for there to be a client and server relationship in all cases (even when only one machine it is client and server on the same machine). The client has a reference to the server, but the client knows no difference between the local reference and server, as CORBA encoding allows for the whole system to operate in a transparent manner.

The IWN engine clearinghouse 30 serves as a XML and CORBA switchboard for the entire system. As noted previously, it consists of an message conversion engine 45, workflow repository 46, data/object store 48 and a transaction-processing engine 47. The Clearinghouse interacts with all external entities via open protocols such as XML and WML, and internal entities via CORBA. CORBA provides a high performance, yet scalable and open infrastructure for IWN engine components to interoperate. Rather than a purely peer-to-peer or purely centralised communications model, IWN engine operates in a hybrid manner. The data/object store 48 can comprise a relational database and OLAP(On line analytical processing) engine suitable for handling dynamic real-time E-commerce transactions and the OLAP of those transactions.

The XML engine clearinghouse 30 provides a core clearinghouse application which interacts with a series of peripheral applications. These include a consumer front end ASP 31, a merchant terminal 32, a merchant partner web frontend 33, an eCommerce brokerage type operation 34, general Internet access 35, a payment engine 36 and a messaging engine 38.

The merchant terminal can comprise a point of sale (POS) terminal running via a browser 40 for operation by a merchant. This terminal can comprise java applets for allowing the terminal to connect to the IWN network by means of a TCP/IP interconnection. This component can be deployed to traditional POS developers as a "developers kit" which they can integrate to a greater or lesser degree into their software, to enable their users to connect to the IWN network. The POS. terminal provides a facility for small to medium sized businesses looking to establish an ecommerce presence. In its simplest form, the POS terminal offers a way to clear plastic and cash transactions via the Internet, superseding traditional EFTPOS. However, NetPOS also allows Enterprise Resource Planning (ERP) point of sale, accounting, inventory management and order fulfilment to browsers, PCs, NCs (Network Computers) and NC based cash register networks 41-44.

Instead of utilizing point-to-point proprietary communication associated with EFTPOS, the system uses the ubiquity of the Internet to carry out transactions. Technically, the relevant API can be implemented as a Java or VB based software application that uses encrypted Internet communications to exchange XML RPC with XMLMarket application services, with, for example, the IWN engine clearinghouse 30. The consumer webfront application service 31 automatically generates an Internet website presence for IWN engine or client's merchants using the information exchanged by devices on the IWN network. The website is a virtual representation of the business, autonomously reachable on the Internet. Consumers can use the website and make purchases without any kind of prior association with an IWN engine clearinghouse 30. Integration between the POS and website transactions via the IWN engine clearinghouse 30 creates a seamless order entry and fulfilment process.

The merchant/partner webfront application service provider 33 is provided to deliver the information to users of the IWN network through any (compliant) World Wide Web browser connected to the Internet. Apart from catering to the smallest enterprises, the merchant/partner webfront ASP provides mobility, deployability and rapid growth capability to businesses of any size.

Both the consumer and merchant/partner ASP are designed to enable conversion of the data to any format, so as to enable it to be viewed on ANY device (ie mobile phone, web TV, PDA, kiosk etc)

A brokerage facility 34 links websites together in a virtual community. The system enables the products of all merchants in the community to be queried in a consistent manner and provide for consumer driven queries which aggregate the data. For example the customer can specify the product and price desired and the database will query all merchants to match the request. Consumers experience a shopping environment, consistent across all vendors within the IWN exchange system. Conversely, other eMalls can present IWN engine generated information or web sites.

Business to business functionality enabled by IWN engine also allow merchants to collaborate for competitive advantage with their major suppliers. IWN engine eMall consumers enjoy an overlay or meta-search engine capable of finding products by multiple criteria across multiple vendors. This is due to the fact that they are able to query not only the IWN engine held by the local client, but also nodes from all over the world.

User, connected by devices and in some cases specifically through browser sessions, are typically only connected on a permanent or semi-permanent basis using low speed lines, whereas the clearing house can be full time connected with high bandwidth, redundant Internet links. Components like the POS devices 40 and brokerage services 34 task the IWN engine 30, which means they can leverage its permanent, high performance capability. The IWN server clearinghouse contains the business logic (described hereinafter) required to initiate and complete transactions, and can interact with any device which transmits standard message formats such as HTML, WML, XML, IMAP etc and is connected to the Internet. It also frees the devices from the requirement to manage multiple connections to multiple parties

Funds transfer within the system is managed a Payment Engine 36. One of the barriers to entry of the existing EFTPOS networks into eCommerce/Internet markets is the proprietary nature of the protocols involved. It is difficult to either effectively integrate these protocols with the more modern worldwide web paradigm into the SME environment, or for the existing financial institutions to develop an alternative to the Internet. XMLMarket merges XML and EDIFACT, (and other standards) by 'front ending' the various proprietary financial networks with an IWN engine.

All IWN network process-to-process communications can take place predominantly on top of the TCP/IP protocol suite. The IWN network preferably conforms to all relevant RFC and (where applicable) ISO standards.

IWN engine is designed to interoperate with the widest variety of organisations possible. To this end, the eXtensible Markup Language, XML, is often used to encapsulate all system-system transactions. XML provides a self-describing document paradigm; these documents are transmitted via HTTP using the XML RPC (XML Remote Procedure Call) standard.

Whenever sensitive or financial data is being transmitted, IWN Engine HTTP application servers uses the Secure Sockets Layer (SSL) protocol to encrypt the data stream. SSL provides strong encryption to protect the data in transit. Participants in the market are pre-screened (to varying degrees depending on their user group) before admission, and authenticated using a non-proprietary protocol such as the RADIUS protocol. Using RADIUS ensures a standards-compliant approach to person-to-business authorisation, and transparently provides the option of using strong cryptographic techniques like challenge-response or one-time password algorithms.

In business-to-business transactions, X.509 certification is used to authenticate automated users transactions, which are also SSL encrypted. X.509 certificates are now available from a number of authoritative sources both internationally and domestically. IWN network centralised and distributed subsystems are protected behind industry standard ICSA

The topology in which IWN network distributed subsystems are deployed for a given region is dependent on demand, bandwidth availability and commercial agreements. From a purely technical perspective, there can be a minimum of one and an arbitrarily unlimited number of each of these distributed processing units. This provides for extremely large-scale deployments.

and retains the ability to deploy in a locally customised fashion where geographical, technical, commercial or political constraints apply. There are three major distributed subsystem orientations, namely customer, vendor and service.

The brokerage WebFront 34 provides a meta-layer on top of arbitrary groups of IWN engine merchants. An IWN engine brokerage differentiates itself from other web-mall offerings by:

-Offering consumers a consistent shopping metaphor overlay, visually consistent across vendors. This is achieved by overlaying a consumer shopping interface with communicates with any IWN engine compliant merchant.

-Single transaction payment services across vendors.

-Cross-vendor searching and comparison (at the vendor's discretion).

-The ability to integrate with and search other popular eMalls in the event of an unsuccessful search.

-An arbitrary number of eMalls across an arbitrary number of vendors.

The IWN engine shopping experience features a customer service window tailored for shopping. Services include gift tokens, lay-buys, recommendations, favourite merchants, and so on. The consumer interface travels with users regardless of the vendor entered. This consistency builds trust, and thereby consumer confidence. A variety of mall concepts is envisaged, with visual, audio and product placement scenarios tailored to target markets.

Workflow operations

The overall operation of the system can be as follows:

1. Transaction enters the system from a remote device.
2. The network processes the request via its IP address and routes it towards the IWN system
3. The IWN system identifies the request using either "tags" in the messaging format (messaging formats may include XML, WML, HTML), IP address, or the port on which the message was received.
4. The IWN system then:
   (a) Identifies workflow implicated by the transaction and begins to execute that workflow:
   (b) Converts the message into a consistent IWN XML DTD to enable the workflow to execute.
   (c). Stores relevant information in the datastore
   (d) Forwards messages to appropriate parties

In many cases "persistent" processes may be frozen in the system to be fulfilled at a later date.

The IWN engine clearinghouse 30 is directly connected to specific providers that are able to fulfil order requests, third party information vendors able to provide information, and service providers that are able to carry out financial transactions such as credit card transactions or the like 36. Also interconnected to the IWN network are suppliers who receive requests for goods, import export agents, warehouse storage facilities and producers. Thereby, the IWN Network is designed to handle logistics and inventory management as well as information about business relationships etc.

The data/object store is interrogated by at least five different sets of user groups (Using the OLAP interface), including web base users, point of sale (NETPOS) users 3-6, product mediators 20-22 and product producers 23, consumers, and the client (telecommunications company or bank). When it is desired to query the database, an OLAP interface carries out a database transaction, returning the results of the query.

In modified embodiments, the number of servers 30 can be replicated. IWN network can then include a series of servers and can be thought of as a series of nodes of different sizes. There may be one or several network operations centres housing all the applications and a master data-store. There are then the client nodes, which are a local replication customised to a demographic and/or political and/or geographical and/or legislative region. These users may be a retail point of sale system, or a larger corporate client.

The global portion (ie either the main network or a node situated at a "client") is differentiated in that all data can be collated and made available through the application server as single source for output in various forms such as HTML, as well as other relevant forms (XML, CMI, WML VRML, SHTML, php, pwp, java objects, etc).

The IWN network is regarded as distributed because clients have relevant and alterable data nodes local to their machines. Hence relevant local and global database updating is required. The IWN network is dynamic because one local or global alteration might create several internal network related operations. The local server receives requests from its local user which will either be on the same machine or though the user's local network. The server will then either get the information from its local data store or download it from the IWN server. The user is able to simply request certain information and the server checks its cache if its there and returns it or otherwise downloads it from the application server (if available) and return and object representing the data.

The local components are only relevant to an individual user and/or user peer groups and is a subset of the larger client database's. Local portions are regarded as local because the portion is local to the user's location.

There is one major authoritative store of information in each region in the servers data/object store. Non global information (like faxes etc) can be independently stored. The global portion has the largest subset of information which is of interest to customers of clients or client peer groups (for example: various on-line web shoppers).

A smaller subset of IWN network information is relevant to a user peer group, and an even smaller subset is of interest to a user. The global portion of the IWN network is kept up to date by periodically up loading local information (this is done when needed, for example: after local alteration). The NETPOS data can be implemented having write through capability, meaning that when a data update is requested by a NETPOS terminal the data will go straight to the main data source (if available) not just the cache. The global portion of the IWN Network also processes information and passes it back to the client, in order to update their local database (again done when needed, for example: after customer alteration). Users access the IWN network using an interface which performs all tasks local to the user. The XML engine 46 runs the queries and return the results to the request source, either the web server or the POS server..

The Following Automatic Functions can also be Provided

(a) Web page creation and on-going maintenance:

The client is able to change a variable in their inventory management system locally. This in turn automatically updates the global portion of the IWN network, which automatically alters relevant web page elements. An example is a change of price on the local machine which is then reflected on the web site.

(b) Ordering and order fulfilment:

The IWN Network has the functionality to allow point to point real time order fulfilment by facilitating catalogue maintenance (though the inventory system dynamically updating the price and the availability of the products) and offering direct connection to a third party order fulfilment client (such as FedEx). This includes the ability to set multiple relationships for each customer. For example supplier A may set different terms of payment and delivery for each customer. The user interface will communicate this order fulfilment criteria and will establish and maintain relationships with each customer. Each user will also have the option of setting automatic re-order points. There re-order points can be set to trigger the dynamic and

continual maintenance of inventory levels in the client's enterprise. The main advantage of this order fulfilment system will be its dynamic nature, for example: a customer queries the central database and assuming availability of the product, results in the order lodgement simultaneously with the order fulfilment client and the product client. The result will be a dynamic update of the level of inventory in the central database.

(c) Accounting:

By utilising the combination of inventory levels with supply side transactions (any transactions involving the movement of goods or services for a whole-sale price, i.e., between clients) and demand side transactions (with a customer) the IWN network has the capacity to provide summary information about the financial position of a client in real time. These reports will be displayed in a number of ways and in accordance with the accounting conventions of the geographical area. It will focus particularly in profits by item type and by supplier (which is a client).

(d) Customer information systems:

By providing basic information about a customer and also about any on-line customers who purchase a show interest in products, the IWN network is able to give the various user groups (Merchants, clients, and consumers, etc) information about demographic factors of their customer base. Using OLAP technology enables unlimited number of views of the data, and resultant queries. It also allows them to track the habits and preferences of consumers and uses intuitive search mechanisms of the network to build personal relationships with the consumer. For example by tracking what a consumer has bought it may suggest other related items or similar items by the same supplier. The key aspect here is that the data can be viewed and modified by the user's customer representative. An application of this is the ability to transact with the on-line customer via chat and other communication means to give detailed information to the on-line customer. This is all done directly from the users local IWN network interface.

(e) Management information systems (particularly inventory):

The IWN network will enable a variety of management information queries to be parsed to the user. An advantage is that the user is able to access information in real time and access information for the local database which has been entered globally. This includes order fulfilment information, information regarding the use and transactions on-line, and all this in combination with traditional point of sale data.

From the user's side, the difference is that the information, once entered in the local database, then periodically updates the global database without any proactive re-entering of information.

The Proactive Functions

The following proactive functions can be provided:

(a) Basic daily transactions and electronic funds transfer:

While the transactions are all being entered in at the local (clients) level, they are also being periodically updated to the global database without being re-entered. This allows the information (local data) to have secure back-up.

The establishment of a secure virtual private network (VPN) will then allow for the free two way transmission of data wherein the local client is able to be updated by the application server of the global database when queries are made to any particular credit verification entity.

Traditionally, funds transfer from the point of transaction (being a retail store, home-office, or back office etc) has been handled by a hardware device commonly called an EFTPOS terminal. Using a compliant point of transaction system, the user is able to clear a transaction directly through their PC without additional hardware. For example, the attached appendix

sets out a point of sale development tool-kit (SDK) called IWNcom which enables transmission of data from a traditional PC to the IWN network for clearance. The data is described in a DTD and thus open to inclusion in a data-store.

The IWN Client SDK (Software Development Kit) is designed to enable third party to integrate their device or applications to take advantage of the IWN network functionality. These SDK's can be Java constructed and hence can readily be ported to the following devices: Windows 95, Windows 98, Linux, Windows 2000, Mac OS, PalmOS, WAP compatible devices, Java Compatible devices, Smart Cards, and Java Cards.

The fundament concept behind the SDK is that it initiates a common set of network level communications, security management, session management, and then transactions over this secure session. Transactions may include (but not limited to):

1. Financial transactions, debit and credit
2. eCommerce transactions (purchasing of products, supply of products etc)
3. Content based information (such as audio and video streaming etc)
4. Bill presentment and payment
5. Electronic ticketing
6. OLAP and other reporting
7. Service provisioning and management (ie activation of new services, termination of services)
8. Smart card re-charging
9. Device management related transactions (such as fault reporting etc)

Technically, the SDK has three layers: Communications, Network Security, and data transmission. The communications layer will typically initiate a network connection , detect status of that connection, and respond to various events relating the connection. The security layer will establish the validity of the user, establish the validity of the server ,and then establish a secure connection between both parties. It may use SLL, DES, TLS, and other encryption and security processes and protocols.

The most complex layer is the data layer, which translates information about the client device, and actions on the client device , into a standard document (usually XML or WML documents) for transmission over the secure network. Information in this document may include information regarding a merchant, consumer, device, products, services, prices, geographical, loyalty.

(b) E-mail 51:

The client is able to send/receive e-mail using the same interface they use for point of sale transactions and all other group one or two functions.

(c) Fax 52:

The client is able to send/receive faxes using the same terminal. These faxes are sent to the IWN network at which point they are distributed to the appropriate recipient. Incoming faxes enter the IWN network and are filed using incoming CLI records.

(d) Accounting:

The client is able to input accounting information and change the price of any resource be it labour, stock item or other and this information is periodically updated to the IWN network Once in the database it can be selectively used for designated areas of the clients web site.

(e) Customer Relationship Management:

The user (local NETPOS terminal user) is able to input customer details and these are updated periodically to the global database for later retrieval if necessary. The local database can draw a series of summary reports from the global database or selected information can be stored locally. Traditionally these queries are referred to as CRM queries

(f) Management information systems(MIS):

A user may take the CRM information and organise it in such a manner as to enable business rules which are particular to their organisation. In such a case they have facilitated a MIS system.

(g) Order and order fulfilment:

The ability to manually enter an order and then have that order updated to the global database to be later verified and placed and then confirmed or alternatively confirmed immediately.

Vendors also uniquely gain synergistic connections with other IWN engine vendors. Through IWN engine, vendors can take advantage of shared warehousing and freight facilities, and economies of scale that they would otherwise not enjoy. Essentially a vendor can automate their production chain using IWN engine as the glue. This can be facilitated in two ways:

1. The IWN engine connects to suppliers directly

2. The IWN client uses established B2B infrastructure and relationships to connect to suppliers. An example is if a telecommunications company or Banking institution has a B2B system which communicated via EDI to a motor car company, IWN engine would simply connect to the in-house application rather than establishing a new connection to the car company.

Although logically one subsystem, the POS interface can be deployed in three distinct configurations. A NC Cash Register System 41, standalone NC/PC system 42, and Merchant/Partner WebFront ASP 43.

Essentially, the IWN server clearinghouse 30 can be unaware of client system topology and the decoupling provided by open standards ensures infinite possibilities for presentation/topology of POS systems into the future. The first two cases, NC Cash Register System 41 and Standalone NC/PC System 42 can be identical technically, differing only in the number of processing units at a customer premises. Inventory, pricing and other operational data is stored locally. Transactions (in both directions) will initially be made using XML, and periodic reconciliation transactions can be performed to ensure convergent views of data at both the clearinghouse and distributed locations.

In the case 40 where a Merchant/Partner is connected using a web browser rather than a dedicated piece of hardware, POS view of is deployed as the Merchant/Partner WebFront ASP 33. The local data for this vendor can be stored within the Merchant/Partner WebFront ASP 33; however this is completely transparent to both the user and the IWN server clearinghouse. The Merchant/Partner WebFront ASP 33 communicates with the IWN server clearinghouse 30 precisely as if it was an NC Cash Register System 41 or Standalone NC/PC system 42 utilising the attached SDK protocol. In all cases, the WebFront provides all required functionality to run the point of sale, inventory, order fulfilment, debtors, and creditors functionality of a SME merchant/partner. Offline contingency capabilities can be provided in the case of dedicated hardware/software.

In some embodiments, a merchant may already be committed to a particular application that cannot interact with XML, and therefore IWN network. Reasons for this can include:

- A closed, proprietary application forces duplicate data entry to other entities.

- The application vendor is not yet committed to XML for business-to-business integration.

- Overwhelming market share by a vendor forces a de-facto standard.

In order to overcome any reluctance to adopt the XMLMarket system, legacy applications 44 can be incorporated into the overall system by deploying a developers kit enabling third parties to develop IWN network extensions to their products, creating IWN network Extended Applications (iEAs). Such an arrangement assists in building a critical mass until XML (or subsequent standards) become a common feature of ERP software.

In an iEA, the functions of the Vendor WebFront are be adapted to the legacy application. Inventory, pricing and other operational data can remain stored locally within the application. Vendor Webfront transactions (in both directions) are made in (at this stage) XML with the IWN server clearinghouse 30, and periodic reconciliation transactions are performed to ensure convergent views of data at both central and distributed locations. To the IWN server clearinghouse 30, the iEA behaves precisely as if it was an NC Cash Register System 41 or a Standalone NC/PC system 42. IWN network extensions are simply used to eliminate redundant data entry, and to ensure synchronisation between the application and the IWN server clearinghouse 30.

The IWN Network Extended Application provides all the required functionality to synchronise the point of sale, inventory, order and fulfilment functionality of a merchant/partner. Offline contingency capabilities are not provided by IWN network; that is they must be provided natively by the legacy application itself.

Payment Engine 36

Current technology for funds transfer, transaction clearing, micropayment services and alternative electronic forms of cash often remain proprietary, non-standard, costly, and difficult to implement. Generally banks and credit providers insist on proprietary physical links with proprietary communications protocols and proprietary POS terminals. Over the Internet, there is currently little improvement with the norm being proprietary technologies, uncertainty, lack of standards and rapid change. One of the major hurdles a small to medium sized enterprise (SME) faces in establishing a web-presence is the requirement to solve all of these problems and to re-implement a financial transaction mechanism. Often, SMEs generally do not have the financial wherewithal to achieve this, resulting in a major barrier to eCommerce adoption. The IWN network solves this problem by abstracting the interface to multiple financial institutions and other payment transaction enablers via the Payment Engine 36. This subsystem provides a CORBA interface to the IWN server clearinghouse 30 through which all funds transfer, credit authorisation, sales transactions and merchant payments and receipts can be processed. The Payment Engine 36 connects to the various financial organisations via the multiple proprietary mechanisms and point to point links, and is able to leverage higher bandwidth connections and better transaction rates due to economies of scale. Financial transactions are simply another form of XML RPC, carried out using SSL encryption and authenticated with X.509 certificates and processed by the IWN server clearinghouse 30. Vendors enjoy significant benefits from this approach, including economies of scale, the ability to adapt to new technologies immediately, automatic transaction dissection and settlement across vendors; and of course, most importantly, simplification. Because the IWN Network paradigm is holistic, then this provides the ability to offer vendors differentiated pricing models for transaction clearance and participation.

In the case where a "Client" has already owns a payment engine (such as the Nobil Gateway produced by Keycorp), the IWN engine can talk directly to the resident payment engine as apposed to the financial institutions.

Messaging Engine 38

Participants in any market require notification, confirmation, information and communication, between consumer and vendor, vendor and vendor, and consumer and consumer. Traditionally, this type of messaging was carried out by telephone

and post, and more recently by manual facsimile. However, the three criteria for effective messaging are timeliness, authenticity and convenience. Telephony provides low authenticity and convenience tests, post provides low timeliness, and manual facsimile can provide low authenticity.

The IWN Messaging Engine 38 provides the IWN server clearinghouse 30 (again through CORBA) with instant capability to generate a message request to any number of participants. Initially, Email 51 and automated facsimile 52 may be provided. Over time, IWN network can introduce further services such as paging, voice synthesis and wireless notification services. All notifications can be serialised and time stamped, allowing a recipient to verify message authenticity as required. Email messages can be digitally signed where financial concerns exist. Finally, participants will be able to nominate preferred mechanisms for contact/notification, increasing the convenience level of the system. As a result, users will enjoy the convenience of integrated messaging from a single point.

IWN server clearinghouse 30

As noted previously, four major conceptual components forge the IWN server clearinghouse Application System. Architecturally, the glue between these components is primarily the Common Object Request Broker Architecture (CORBA). CORBA provides the flexibility required to implement a scalable solution at a single point: the IWN server clearinghouse may run on a minimum of one CPU, or may be distributed over multiple CPUs and multiple devices. If necessary, the IWN server clearinghouse can be a singular entity for a given IWN eXchange, and multiple Exchanges can seamlessly interact using XML RPC. The most immediate consequence of this is that inter-market e-fulfilment chains are entirely possible. A worldwide chain of IWN eXchanges serving country or continental regions can then be constructed, permitting global shopping with local supply. The clearinghouse elements include:

Data/Object store 48

The foundation of the IWN server clearinghouse 30 is a large scale relational database management system with an object oriented paradigm overlay. Consumer and vendor information is stored safely and securely within the Data/Object store 48, and is subject to inherent encryption and access control facilities. Database design can accommodate international, multi-lingual information, incorporate audit trails and implement two-phase commit technology ensuring reliability of transactions. Vendors, participating as merchants or partners can maintain their own local data stores. These can be bi-directionally synchronised via XML with the Data/Object store. Because the Data/Object store is relational, IWN network offers deeply flexible reporting and analysis opportunities to participants.

XML Engine 46

XML can be the primary messaging service used in the system. As such a component which addresses this component is part of the architecture. As new standards develop, this engine will be extended to include emerging standards. One of the primary benefits of XML is that it provides a common language for structuring documents that flow between business partners in a supply chain. Since XML documents share a common structure, the process of transforming documents into new data representations is vastly simplified. While XML offers far more flexibility and extensibility than either traditional messaging or EDI, XML by itself does not deliver the level of integration required to implement the IWN network. The XML Engine 46 adapts other subsystems to communicate using XML, and insulates them from the processes of encoding and decoding XML. It provides the infrastructure to manage the integration process over time, securely and reliably route requests, and translate between messages that conform to different Document Type Definitions (DTDs). It also provides the facilities for:

- Integration of heterogeneous systems across firewalls.
- Authorisation and security across corporate firewalls.
- Provide guaranteed delivery of messages.

- Support for both document and service based integration.
- Change management and support for ongoing evolution.
- Support for XML-based vocabularies and technologies.
- Data transformation and mapping.
- Scalability and performance issues.

Workflow repository 45

Workflow is concerned with providing the information required to support each step of the business cycle. The IWN network involves a mesh of information and transaction flows between market participants. In order to manage these flows, the business rules for each transaction and participant type are encoded and stored within the Workflow Repository 45, a logical entity implemented on top of the data/object store. It combines rules, which govern the tasks performed, and coordinates the transfer of the information required to support these tasks Workflow Repository tasks may be physically moved over the network between participants or maintained in the IWN server clearinghouse with the appropriate processes given access to the data at the required times. Triggers are implemented in the system to escalate exception conditions in the event of problems within the system.

The operation of the workflow is depicted in Fig. 2. Data 60 sent to the IWN clearinghouse server, if not already in an XML format is forward to a translator 61 for translation into an XML format 62. The XML document is then queued 45 in the workflow repository. The workflow repository loops through a process of requesting document objects 64, matching the document against possible types 65, determining what corresponding workflow should be undertaken for the document type 66. The workflow is built 67 and added 68 to the workflow queue. Each added portion of workflow is then process from the queue 69 and eventually a response returned 70.

Hence, the workflow is spawned from the contents of each XML document. Once the document has been revived and converted into a consistent data type, the engine processes the document based on actions relating to XML definition in each document.

The first object that is requested is a "Profile" that matches the IWN exchange user with a profile. This profile is then matched with elements in the data and from this a workflow is spawned. This workflow is comprised on a number of unique "actions" that together form a "graph" that define a particular route for each workflow.

Transaction processing engine 47

The IWN network employs a standard 3-tier client/server application model. Server applications can be written as a set of interoperable, modular components using standard computer languages — such as C, C++, and Java. The Transaction Processing Engine then runs them in its scalable, high-performance, secure, transactional environment.

The Transaction Processing Engine 47 logically sequences interactions, using business rules from the Workflow Repository 45 to process transactions from the XML Engine 46, updating the Object/Data Store 47 along the way. It ensures that the business transaction is secure and reliable, and is completed with integrity. Some of the capabilities the Transaction Processing Engine brings to the IWN Network include:

- Distributed Transaction Management - servers can participate in a distributed transaction that involves coordinated updating of multiple databases. The Transaction Processing Engine's transaction management helps ensure that all databases are updated properly, or will rollback the databases to their original state, assuring that data integrity is maintained despite component failures.

- Transaction Queuing - provides flexibility in how and when transactions should be processed or deferred.

- Event Brokerage - allows for posting of system or application events that can be subscribed to by any authorized application component in the system.

From the above description and client specifications. the following product functions may be identified:

The IWN server clearinghouse handles the following transactions: Transaction verification; Modify web-store information: Push / pull merchant website information; Modify inventory levels; Serving of ASPs: Issue Mime. SSL; Queue messages: Prepare reports: Transmit payment receipt numbers; Send lay-buy information to Netpos; Transmit purchase requests to POS.

The IWN server clearinghouse deals with the following transaction interactions with the Payment Engine: Credit card verification: Currency queries; Debit information.

The IWN server clearinghouse processes information from the Consumer Web-Front ASP. This includes sorting of information: building the various websites and payment methods

The IWN server clearinghouse assists in the operation of the E-Brokerage ASP in addition to the Consumer ASP in the following manner: Provides an access point for a customer; Run intelligent queries; Store customer preferences; Allow dispute resolution; Payment methods

The IWN server clearinghouse also allows third-party data interpretation

- Logistics companies interactions.
- Integrate other multinational companies.
- Integrate other e-commerce programs.
- Integrate ERP. SAP

The IWN server clearinghouse must be able to handle:

- Product information
- Merchant information
- Supplier / mediator / customer information
- Limited website information
- Financial (goods) information
- Transactions (+, -, *, /)
- Receipts
- Tax
- Payment method selection
- Lay-buy information
- Show customer information
- Show shipping information
- Quotes

- Terms of sale
- Price specials, promotions
- Dispute reason codes
- E-mail: Send, receive, history
- Business reports: Sales (per period) customer, supplier, product, employee, inventory, delivery, profit.
- Accounting information: Accounts receivable, payable, total labour costs, cost of goods sold, etc.
- Faxes
- Web page information: (Wizard setup, Product categorisation, Webpage administration, Statistics)

The following describes the users interacting within the system.

Consumers

A consumer is simply defined as a person or party that initiates a transaction with a vendor or multiple vendors within IWN network. Internet consumers use an brokerage or a merchant site created by the IWN Webfront ASP 31 as a user interface through which transactions are performed. Of course, consumers can also initiate transactions physically at the customer premises. Customers are also able to spawn their own personal IWN customer interface which is customised to their needs and allows them to conduct specified OLAP queries of the database (see diagram x – peter ask me for the customer details screen)

Merchants

Individuals or organisations that offer products or services for sale to consumers are defined as Merchants within the IWN network framework. They participate in transactions using a spectrum of equipment, from sophisticated proprietary software with XML capabilities through to a simple web browser. All transactions, whether cash or plastic, are recorded and tracked on behalf of the Merchant by an instance of the IWN engine thereby allow management information reporting

Mediators

A mediator is a merchant that sells to other merchants. They may offer semi-finished goods or components (traditional supplier role), or services that complement or facilitate the production process. A freight company, insurance agent, graphic designer, vegetable wholesaler or coffee bean supplier may be considered an IWN eXchange product mediator. Common mediators include distribution, warehousing and import/export agents.

Enablers

Financial institutions, credit providers, cyber-cash agencies, micro-payment vendors, telecommunications carriers, software and web design companies are all enablers within IWN network. An enabler provides services to consumers, merchants, and mediators by facilitating transactions as opposed to supplying the goods and services that comprise them.

Other Markets

The IWN network is based on the open, scalable, business-to-business standards such as XML. Because of this, the IWN network can communicate with any other market or party that also adopts open standards. For example, Microsoft is promoting a technology called "BizTalk", which consists of a set of XML DTDs for business-to-business transactions. IWN network is automatically compatible with BizTalk as a result.

Clearing House Functional requirements

The following tables set out in more detail the implementation of the functional requirements of the overall IWN server clearinghouse system.

Operation	**Transaction verification**
Inputs	• Transmitting purchase request
Processes	• Any purchase or return of item will be logged in the system either by a user input or automated logical process • Verification of user. • Verification of the amount being sent. • Identification of the debtor and creditor • Alert to the payment engine to transfer funds from purchaser to vendor
Outputs	• Alert / confirmation to the user of the transaction • Logistics / delivery information • Transaction / order number • Suggestive sale information

Operation	**Modify web-store information (From POS)**
Inputs	• Product descriptions. • Product Id's. • Product pricing information. • Product availability. • Product preference and placement on web page and in terms of search queries and results • Modify merchant information center • Modify credit card acceptance • Modify store status (open/closed/coming soon) • Unique product placement circumstances such (ie Gift opportunities) • Product logistic information. • Product compatibility information (product inclusive and exclusive rules).
Processes	• On upload of product information into the database, and after verification acceptance, the above information is stored in the database

Outputs	• Confirmation of product catalogue entry. • Acceptance of product into system

Operation	**Push / pull merchant website information**
Inputs	Upon request of a URL by a user or need for additional information by the clearing house to parse to the web front
Processes	• User request of URL establishes call for web front information • If customer has established profile, log of information • Regardless of customer profile, log of all URL usage information • Based on customer profile, query currency, language and other demographically specific deliverable's • Query of relevant merchant information for changes at NetPOS level
Outputs	Presentation of URL along with information builds web front and marketing information. The web front is displayed in the relevant dialect and currency for the user

Operation	**Modify inventory levels**
Inputs	• Inventory information • Availability information • Stock level information • Deliverability information • Backorder fulfilment information
Processes	• Transaction at POS or Web Front sends product update to clearing house. • Clearing house checks for stock availability and for the validity of transaction • Transaction is approved or rollback occurs
Outputs	• updated inventory levels

Operation	Serving of ASPs
Inputs	Request for an application by user at an agreed fee and duration
Processes	User requests application through any compliant device Clearing house queried for availability

	IWN server clearinghouse evaluates bandwidth availability Time stamp establishes the start time of use Fee is added to user account
Outputs	Application delivered for use and session spawned for pre-agreed period User is charged

Operation	**Issue Mime, SSL**
Inputs	Secure session requested by user
Outputs	Secure session is spawned for user

Operation	**Queue messages**
Inputs	Any transaction requested from the IWN server clearinghouse
Processes	User makes request for URL. application. information. or intuitive response from the IWN server clearinghouse
Outputs	Response of success or failure from the server

Operation	**Prepare reports**
Inputs	User requests information from the server pertaining specifically to statistical or aggregated
Processes	User authenticated User privileges established with regards to the requested information. Success or failure for request Report generated from relevant data
Outputs	Report delivered to web front or Netpos Upon failure rollback

Operation	**Interface with third party vendors**
Inputs	Request from third party vendors for information Request for information from third party vendors
Processes	Third party sends query to IWN server clearinghouse via either dedicated line or TCP/IP request Query workflow repository for data type recognition

	If data type recognised, converted to standard DTD User allocated access privileges Request for information processed and checked against access rights Submission of information processed and stored Persistent object created if request for later fulfillment Request to vendor fulfilled
Outputs	Information from vendor fulfilled/rejected Request from vendor fulfilled/rejected Request to vendor fulfilled/rejected

Operation	**Send lay-buy information to Netpos**
Inputs	Entry of credit card information. Debtor information and verification (credit card user) Debtor contact information Product information. Pricing information. Product availability and promotion information.
Processes	Creation of a lay-buy account. Linking of current to previous lay-buy accounts. Setting of lay-buy time periods
Outputs	Lay-buy confirmation information Transmittal of lay-buy status Transmittal of lay-buy costs and expenses

Operation	**Transmit purchase requests to Netpos**
Inputs	Item purchased Debtor information Payment method Credit card details Logistic information (delivery schedules and methods)
Processes	Customer purchases item from any device

	Request sent to clearing house Replication of data at clearing-house and NetPOS
Outputs	Confirmation of purchase Receipt number Delivery details

Operation	**Verify credit card information**
Inputs	• User details • Credit card details • Purchase details • Transaction confirmation details • Receipt number
Processes	• Credit card information stored in clearing house • User authorised at log-in • Purchase request processed against inventory levels • Transaction sent to acquiring bank for verification • If successful changes to inventory levels • If fail then rollback transaction
Outputs	• Verification of credit card details

Operation:	**Transmittal of payment receipt numbers**
Inputs	• Account / transaction number • Amount of transaction • Method of payment • Date of payment • Payer information (user)
Processes	• Update of user account information.
Outputs	• Transaction amounts due • Receipt number

Operation:	**Currency conversion**
Inputs	• Merchant currency preference • User currency preference • Current currency information
Outputs	All users view currency that they have previously selected

Operation	**Administration reports**
Inputs	Information stored by the server will be used to determine the number of users.
Processes	As part of the report generation process – the number of users that have contributed information will be determined from information stored on the server.
Outputs	The number of users who contributed will be placed at the top of the summary report for administration purposes

User Interfaces

Apart from the usual browser interfaces, the POS interface for can be a point of sale hardware device such as the Comm2000 device from Keycorp. The configuration at point of sale may include a cash register, customised keyboard and other typical point of sale devices.

As IWN network is based on a client/server computing environment, the necessary hardware device required to access the network is a Internet connection to connect to the IWN server clearinghouse server. This can be implemented through a third party ISP.

Software Interfaces

The software to be interfaced with the IWN network can be a Java client running on a Web browser (Netscape Navigator 3.x, Netscape Communicator 4.x, Internet Explorer 3.x). Alternatively, the operating system (Windows 9x, Windows NT, Unix, MacOS) can interact directly using the TCP/IP protocol.

As described above, any third party can interact with the system by integrating to it using the IWNCom's development kit as shown in the attached appendix A.

(c) Communications Interfaces

The data communication protocol required to use the IWN Network can be the standard TCP/IP protocol. The type and speed of connections to the Internet will be varied amongst the different users and does not impact the design of the system.

POS to XML

The POS device provides a direct translation into XML. Initially, the POS device opens a TCP/IP connection with the IWN server. The credit or debit card data is then sent to the server. The process firstly involves the negotiation of connection and session's with the IWN engine. The process involved is slightly different for Credit and Debit. For a credit card, the steps include:

1.. Merchant prompted to swipe card either by the third party software (such as Quicken or MYOB) or by the IWN client side component directly

2. Merchant swipes card

3. Card details sent back to the PC (instead of remaining in the EFTPOS device)

4. Either information gathered by the third party software and passed to the IWN object or gathered by the object directly

5. IWN Client side Component (either Java based or ActiveX) collects the credit information converts it to AS2805 compliant XML data, and sends it to the server. The collection of information can also include (not limited to) product information , merchant information, and information about the terminal. The product information is extrapolated from a local data store using either supplied APIs or via customised integration. Alternatively the product information can be sent as a batch file at pre-determined intervals.

6. The server sends a response back to the client side component which leads to receipt printing and other related retail business requirements.

Debit solution from POS

The debit solution may include more traditional pin key encryption devices (such as the Com2000 device from Keycorp), or newer technologies such as WAP and Bluetooth. Using the more traditional POS hardware configuration, the system can operate by the following steps:

(a) Merchant prompted to swipe card either by third party software (such as MYOB or Quicken) or by the IWN client side component:

(b) Merchant swipes card:.

(c) A ECR message sent to the EFTPOS Hardware (Which is capable of Key PIN Encryption). (d) The PIN Pad or like device prompts the user to enter a PIN number (usually four digits). The user enters the PIN number

(e) The PIN number is encrypted with 3DES encryption and set back to the computer via an ECR Message. The message is either received by the third party application or the IWN client side component. The 3DES encrypted message is then sent to the IWN server along with other XML information regarding the transaction including (but not limited to) Product, customer, merchant, and terminal information.

(f) The IWN Server passes the PIN to the Acquiring institution (potentially via an internal gateway first) and then through to the issuing bank, the issuing bank then returns a response as to the funds available.

An alternative method can comprise the following steps:

1. Mercant Swipes card in POS terminal

2. POS encrypts the data (DES usually) and sends it back to the PC

3. Only the AS2805 (or similar financial information) is sent to either:

(a) The IWN engine which then forwards it to a financial institution

(b) Directly to the financial gateway

4. Then when the message is confirmed and sent back to the server a new XML document is spawned that sends the transactions details to the IWN server (over the same link)

It is also important to identify that information is only stored in the IWN server once the transaction has been approved (besides fault logging and invalid transaction logs).

The debit transaction can be extended to other arrangements which utilise non traditional Point of Sale Hardware. For example, debit card transactions can be conducted over a mobile phone interface using a WAP enabled phone having Key PIN encryption. The process, as illustrated in Fig. 3 can proceed as follows:

1. Sale transaction agreed upon between Merchant/Retailer 71 and customer 72;
2. Merchant selects debit card on their POS device 73;
3. Merchant enters mobile phone number or initiate SMS message on the POS device;
4. Message sent to WAP or SMS server 74;
5. Message sent to IWN Engine 75;
6. Message forwarded to mobile phone 77 via WAP Gateway 74;
7. PIN entered on mobile phone and encrypted and sent to IWN engine 75;
8. IWN Engine sends encrypted PIN to Acquirer Gateway 78 and Sends Merchant ID to the IWN Engine;
9. IWN Engine waits for response;
10. Acquirer 78 forwards the PIN to the card issuer 79;
11. PIN authenticated by the Issuer 79;
12. Response sent to IWN Engine 75;
13. Response forwarded to merchant point of sale 73;
14. Sale completed or denied.

Alternatively, the system can operate to include consumer initiated transactions over WAP. This can comprise the steps of:

1. Sale completed
2. Merchant selects Debit via WAP/SMS option
3. Consumer Selects Debit via WAP/SMS
4. Consumer enters Merchant ID via WAP interface of mobile 77;
5. Consumer enters PIN via WAP interface;
6. Consumer sends message with PIN (Key Pin encrypted) to the IWN Engine 75;
7. IWN Engine sends encrypted PIN to Acquirer Gateway 78 and Sends Merchant ID to the IWN Engine
8. IWN Engine waits for response
9. Acquirer forwards the PIN to the issuer 79;
10. PIN authenticated by the Issuer 79;
11. Response sent to IWN Engine 75
12. Response forwarded to merchant point of sale73

13. Sale completed or denied

Alternatively, the system can operate to include merchant and consumer initiated transactions a Bluetooth network. For example, where the merchant has a blue tooth interface and the user has a Bluetooth enabled phone with a Key PIN encryption device, the process for executing a sale can comprise:

1. Sale completed
2. Merchant selects Debit via Bluetooth option
3. Merchant swipes card
4. Card and Merchant details sent via Bluetooth to the phone
5. Consumer enters PIN
6. Consumer sends message with PIN (Key Pin encrypted) to the WAP Gateway
7. WAP Gateway sends PIN & card number to IWN Engine
8. IWN Engine sends encrypted PIN to Acquirer Gateway and Sends Merchant ID to the IWN Engine
9. IWN Engine waits for response
10. Acquirer forwards the PIN to the issuer
11. PIN authenticated by the Issuer
12. Response sent to IWN Engine
13. Response forwarded to merchant point of sale
14. Sale completed or denied

Further, where the consumer wishes to initiate a Bluetooth transaction, the transaction can be as follows:

1. Sale completed
2. Merchant selects Debit via Bluetooth option
3. Consumer enters PIN into mobile
4. PIN encrypted in SIM card
5. Card details sent via Bluetooth to the POS device
6. Merchant swipes card
7. POS device sends PIN & card details to IWN server
8. IWN Engine sends encrypted PIN to Acquirer Gateway and Sends Merchant ID to the IWN Engine
9. IWN Engine waits for response
10. Acquirer forwards the PIN to the issuer
11. PIN authenticated by the Issue
12. Response sent to IWN Engine
13. Response forwarded to merchant point of sale

14. Sale completed or denied

Graphical User Interface (GUI) to the IWN Clearinghouse engine

As each transaction can be stored in the data/object store, a complete overview of each independent IWN Engine system can be made available from a GUI administration window. The views provided preferably include:

1. Service activation
2. Service Provisioning
3. Work-flow manipulation
4. Logical network overviews
5. View of all "actions" available in the system
6. View and manipulation of work-flow "graphs" representing the logical order of actions in forming any one work-flow
7. View of devices on the network and the role that they play in the network overall
8. Activation of new devices
9. Activation of new clients and related business requirements
10. Activation of new client types (such as an SME Grouping)
11. View of logs for each client, device, service
12. Mapping of new datatypes
13. Mapping of existing datatypes to new services or devices
14. Automated audit trails
15. Activation of peering to other engines

OLAP Processing

The data in the data/object store repository can be queried using OLAP techniques (online analytical processing), ROLAP (Relational Online Analytical Processing), MOLAP (Multi-dimensional on-line analytical processing) , and other similar techniques. OLAP (online analytical processing) enables a user to easily and selectively extract and view data from different points-of-view. Hence, this provides an extremely flexible query tool that enables multi-dimensional queries. OLAP uses a multi-dimensional data base where each data element is stored as a highly discrete piece of information. OLAP can find the intersection of two to "n" relationships between the data.

The IWN architecture uses the following techniques to extract information from the data-store:

1. Any device or access point either compatible with or understood by the IWN engine (in terms of data and network compatibility) sends a query to the associated server (ie WAP server, HTTP server) or connects directly to the IWN engine.
2. The message format is converted to the appropriate data type for the IWN engine
3. The message is recognised by the Engine as an OLAP query

4. This spawns a connection to the IWN Servlet Engine which launches a servlet to mange the query
5. The servlet then sends all queries to the Workflow Repository Manager that protects the IWN data-stores from external tampering
6. The Workflow Repository Manager connects to the data base and returns all relevant information to the Servlet
7. The Servlet then returns the response to the Engine and the transaction is processed and sent back to the appropriate device

The OLAP architecture is novel in that it deals with ubiquitous devices and multiple queries from any of, but not limited to, the following: Web Browsers, WAP Devices, WebTV Devices, Java enabled devices (Ie that can run a JVM), Kiosks, ATM's (Automated Teller Machines). The OLAP architecture can be designed to accept queries from any device (Sending valid requests) and return response to that said device (using valid responses).

Peering multi-instances of the engine

Distributing various instances of the engine allows each engine to link together and appear to be one engine under particular circumstances. This enables two owners of the engine to enter into commercial agreements to contribute client collateral to increase the momentum of their commercial endeavours. For example, if company A has 100 merchants using their engine and company B has 100 merchants, they are able to peer their engines to enable any consumer seeking merchants to query all 200 merchants.

This process can be enabled via a secure socket connection between the engines and the ability for a OLAP query on one engine to access a business rule to query another engine/s and the IP address of the other engine/s. Much like a search engine, the Servlet technology then opens a session with the foreign engine and awaits the complete query of all local and foreign engines before returning the response to the user.

Obviously various modified embodiments and alternative representations are possible. For example, in Fig. 4 there is illustrated an alternative representation of an embodiment of the present invention. Various interactive devices, for example shop front point of sale terminals 80, Internet device 81, WAP devices 82, Kiosk terminals 83, PDAs 84 and Web TV devices 85 are each equipped with an Sdk to convert transactions into an XML document in accordance with associated document type definitions. The XML documents are sent to the XML engine 86 for action and storage. The XML engine in turn provides a series of modules for interaction with the data store. These include an OLAP module 87, logistics module 88, client billing system 89, loyalty modules 90 and brokerage portal 91 and bank clearing house 91. Further, in various embodiments, the IWN engine 86 can be paired with other engines 94 and undertake other transactions with legacy applications and other internet system 95. Fig. 5 illustrates an alternative architectual arrangement and Fig. 6 illustrates the processing of a transaction in accordance with the aforementioned principles, with Fig. 7 illustrating an alternative view of the architecture of the IWN Engine.

It would be appreciated by a person skilled in the art that numerous variations and/or modifications may be made to the present invention as shown in the specific embodiments without departing from the spirit or scope of the invention as broadly described. The present embodiments are, therefore, to be considered in all respects to be illustrative and not restrictive.

Appendix A

Login DTD

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<!-- saved from url=(0063)http://dev.indwide.net/milestones/milestone-
4/xml/dtd/login.dtd -->
<HTML><HEAD>
<META content="text/html; charset=windows-1252" http-equiv=Content-Type>
<META content="MSHTML 5.00.2919.6307" name=GENERATOR></HEAD>
<BODY><XMP><!--
      IWN Login DTD
      Version: $Id: login.dtd,v 1.4 2000/06/05 03:04:46 dth Exp $
      Milestone: 4
-->

<!ENTITY % iwn_elements                "login">
<!ENTITY % login_elements        "(request | response)">
<!ENTITY % request_elements            "authentication">
<!ENTITY % response_elements           "authentication?, code, description,
options?">
<!ENTITY % authentication_elements     "(id, password?) | dsa">
<!ENTITY % options_elements            "workflow+">

<!ELEMENT iwn       (%iwn_elements;)>
<!ATTLIST iwn           version CDATA #REQUIRED>
<!ATTLIST iwn           session CDATA #IMPLIED>

<!ELEMENT login           (%login_elements;)>

<!ELEMENT response  (%response_elements;)>
<!ELEMENT request   (%request_elements;)>

<!ELEMENT authentication   (%authentication_elements;)>
<!ATTLIST authentication   type CDATA #REQUIRED>
<!ATTLIST authentication   algorithm CDATA #IMPLIED>
<!ELEMENT id        (#PCDATA)>
<!ATTLIST id type CDATA #IMPLIED>

<!ELEMENT password  (#PCDATA)>
<!ELEMENT dsa       (#PCDATA)>

<!ELEMENT code          (#PCDATA)>
<!ATTLIST code          type CDATA #REQUIRED>
<!ELEMENT description   (#PCDATA)>

<!ELEMENT options   (%options_elements;)>
<!ELEMENT workflow  (#PCDATA)>
<!ATTLIST workflow  id CDATA #REQUIRED>

</XMP></BODY></HTML>
```

XML Login Request

```
<?xml version="1.0" ?>
<!DOCTYPE iwn (View Source for full doctype...)>
 <!-- Comment-->
 <iwn version="0.4">
 <login>
 <request>
 <!--
 IWN Users can login either by their email address
or their IWNUserID.
example:
<id type="email">foo@bar.com</email>
<password>foo</password>
or
<id>959818</id>
<password>foo</password>
-->
 <authentication type="iwnuser">
<id type="email">bottlefast@telstra.com</id>
<password>merchant</password>
</authentication>
```

```
</request>
</login>
</iwn>
```

XML Login Response

```
<?xml version="1.0" ?>
<!DOCTYPE iwn >
 <!-- comments -->
 <iwn version="0.4" session="6d518aa77a49d8bc823037111dd877d6931d6245">
 <login>
 <response>
<code type="login">0</code>
<description>Login Successful</description>
 <!--
      These are the workflows that the user is allowed to execute.
      In this case, the user can now
      - Request a Purchase ID
      - Request a Transaction Status
      - Logout
-->
 <options>
<workflow id="2">Logout</workflow>
<workflow id="3">Purchase ID</workflow>
<workflow id="7">Transaction Status</workflow>
</options>
</response>
</login>
</iwn>
```

Purchase DTD

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<!-- saved from url=(0066)http://dev.indwide.net/milestones/milestone-
4/xml/dtd/purchase.dtd -->
<HTML><HEAD>
<META content="text/html; charset=windows-1252" http-equiv=Content-Type>
<META content="MSHTML 5.00.2919.6307" name=GENERATOR></HEAD>
<BODY><XMP><!--
      IWN Purchase DTD
      Version: $Id: purchase.dtd,v 1.4 2000/06/05 03:04:46 dth Exp $
      Milestone: 4
-->

<!-- <iwn> children -->
<!ENTITY % iwn_elements                "purchase">

<!-- <purchase> contains either <request> or <response> -->
<!ENTITY % purchase_elements           "request | response">

<!-- <request type=".."> -->
<!ENTITY % request_types         "purchase | purchaseid | purchasereversal |
purchaseauthorize">

<!-- <request type="purchase"> children -->
<!ENTITY % purchaserequest_elements    "source, delivery, payment, products?, cost">

<!-- <request type="purchasereversal"> children -->
<!ENTITY % purchasereversal_elements   "purchaseid">

<!-- NOTE: purchaseid and purchaseauthorize have no children other than
<authentication> -->
<!--       so they are not specified here, as <authentication> is a required element
-->

<!-- <authentication> children -->
<!-- NOTE: digital signatures/hashes would be added here -->
<!ENTITY % authentication_elements     "(id, password?) | dsa">

<!-- <response> children -->
<!ENTITY % response_elements           "authentication?, code, description, rrn?,
options?">

<!-- <options> children -->
<!ENTITY % options_elements            "workflow+">

<!-- <payment> children -->
<!ENTITY % payment_elements            "card | iwnuser">

<!-- <source> children -->
<!ENTITY % source_elements       "pos | web">

<!-- <delivery> children -->
<!ENTITY % delivery_elements           "address | pos">

<!-- <card> children -->
<!ENTITY % card_elements         "name?, number, expiry, account">

<!-- <products> children -->
<!ENTITY % products_elements           "product*">
<!ENTITY % pos_elements                "terminalid">

<!-- <source/pos> children -->
<!ENTITY % web_elements                "client, server">

<!-- <delivery/address> children -->
<!ENTITY % address_elements            "level?, number, street, suburb, city, state,
country, postcode">

<!-- <products/product> children -->
<!ENTITY % product_elements            "productid, name?, description?, unit,
quantity, cost*">
```

```
<!-- <products/product/quantity> children -->
<!ENTITY % quantity_elements          "cost*">

<!ELEMENT iwn       (%iwn_elements;)>
<!ATTLIST iwn       version CDATA #REQUIRED>
<!ATTLIST iwn       session CDATA #IMPLIED>

<!ELEMENT purchase  (%purchase_elements;)>
<!ATTLIST purchase  id CDATA #IMPLIED>
<!ATTLIST purchase  authorize CDATA #IMPLIED>

<!-- <request> elements -->
<!-- <request> must contain <authentication>, and either purchase reversal children
or purchase request children -->
<!ELEMENT request   (authentication, ((%purchasereversal_elements;) |
(%purchaserequest_elements;))?)>
<!ATTLIST request   type (%request_types;) #REQUIRED>

<!-- <request/authentication> elements -->
<!ELEMENT authentication  (%authentication_elements;)>
<!ATTLIST authentication  type CDATA #REQUIRED>
<!ATTLIST authentication  algorithm CDATA #IMPLIED>
<!ELEMENT id        (#PCDATA)>
<!ATTLIST id    type CDATA #IMPLIED>

<!ELEMENT email           (#PCDATA)>
<!ELEMENT password  (#PCDATA)>
<!ELEMENT dsa       (#PCDATA)>

<!-- <request/purchaseid> elements -->
<!ELEMENT purchaseid      (#PCDATA)>

<!-- <request/source> elements -->
<!ELEMENT source    (%source_elements;)>

<!-- <request/source/pos> elements -->
<!ELEMENT pos       (%pos_elements;)>
<!ELEMENT terminalid      (#PCDATA)>

<!-- <request/source/web> elements -->
<!ELEMENT web       (%web_elements;)>
<!ELEMENT client    (#PCDATA)>
<!ELEMENT server    (#PCDATA)>

<!-- <request/delivery> elements -->
<!ELEMENT delivery  (%delivery_elements;)>

<!-- <request/delivery/address> elements -->
<!ELEMENT address       (%address_elements;)>
<!ELEMENT street    (#PCDATA)>
<!ELEMENT suburb    (#PCDATA)>
<!ELEMENT city            (#PCDATA)>
<!ELEMENT state           (#PCDATA)>
<!ELEMENT country   (#PCDATA)>
<!ELEMENT postcode  (#PCDATA)>

<!-- <request/delivery/pos> elements -->
<!-- <!ELEMENT pos         (#PCDATA)> -->

<!-- <request/payment> elements -->
<!ELEMENT payment   (%payment_elements;)>

<!-- <request/payment/card> elements -->
<!ELEMENT card            (%card_elements;)>
<!ELEMENT name            (#PCDATA)>
<!ELEMENT number    (#PCDATA)>
<!ELEMENT expiry    (#PCDATA)>
<!ATTLIST expiry    month CDATA #REQUIRED>
<!ATTLIST expiry    year CDATA #REQUIRED>
<!ELEMENT account   (#PCDATA)>

<!-- <request/products> elements -->
```

```
<!ELEMENT products  (%products_elements;)>

<!-- <request/products/product> elements -->
<!ELEMENT product   (%product_elements;)>
<!ATTLIST product   merchantid CDATA #REQUIRED>
<!ELEMENT productid (#PCDATA)>
<!ATTLIST productid type CDATA #IMPLIED>
<!ELEMENT description      (#PCDATA)>
<!ELEMENT unit             (#PCDATA)>
<!ATTLIST unit             name CDATA #REQUIRED>

<!-- <request/products/product/quantity> elements -->
<!ELEMENT quantity  (%quantity_elements;)>
<!ATTLIST quantity  number CDATA #REQUIRED>
<!ELEMENT cost             (#PCDATA)>
<!ATTLIST cost             currency CDATA #REQUIRED>
<!ATTLIST cost             name CDATA #REQUIRED>
<!ATTLIST cost             rate CDATA #IMPLIED>

<!-- <response> elements -->
<!ELEMENT response  (%response_elements;)>
<!ELEMENT code             (#PCDATA)>
<!ATTLIST code             type CDATA #REQUIRED>
<!--
<!ELEMENT description      (#PCDATA)>
-->
<!ELEMENT rrn              (#PCDATA)>

<!-- <response/options> elements -->
<!ELEMENT options   (%options_elements;)>
<!ELEMENT workflow  (#PCDATA)>
<!ATTLIST workflow  id CDATA #REQUIRED>

</XMP></BODY></HTML>
```

Purcase ID Request

```
<?xml version="1.0" ?>
<!DOCTYPE iwn >
 <!-- comment-->
 <iwn version="0.4" session="6d518aa77a49d8bc823037111dd877d6931d6245">
 <purchase>
 <request type="purchaseid">
 <authentication type="iwnuser">
<id type="email">bottlefast@telstra.com</id>
<password>merchant</password>
</authentication>
</request>
</purchase>
</iwn>
```

Purchase ID Response

```
<?xml version="1.0" ?>
<!DOCTYPE iwn >
 <!-- comment -->
 <iwn version="0.4" session="6d518aa77a49d8bc823037111dd877d6931d6245">
- <!--  specifies the id of the purchase -->
 <purchase id="acs233d23dacad">
 <response>
<code type="purchase">128</code>
<description>Purchase ID Successfully Assigned</description>
 <options>
<workflow id="2">Logout</workflow>
<workflow id="4">Purchase</workflow>
<workflow id="7">Transaction Status</workflow>
</options>
</response>
</purchase>
</iwn>
```

Purchase request

```
<?xml version="1.0" ?>
<!DOCTYPE iwn (View Source for full doctype...)>
 <!-- comment -->
 <iwn version="0.4" session="6d518aa77a49d8bc823037111dd877d6931d6245">
 <!--

      if this purchase should be automatically processed
      set the authorize flag to true.
      from a POS unit this will almost always be true,
      but from a web client, the authorization will
      most likely be false, to allow the merchant to
      manually authorize the purchase via email/messaging.
-->
 <purchase id="acs233d23dacad" authorize="true">
 <request type="purchase">
 <!--
                    AUTHENTICATION
-->
 <authentication type="iwnuser">
<id type="email">bottlefast@telstra.com</id>
<password>merchant</password>
</authentication>
 <!--
 SOURCE
Valid Types: web, pos
the source of this document
-->
 <source>
 <pos>
<terminalid>12346</terminalid>
</pos>
 <!--
If source is web, <client> specifies the
customers IP address. <server> indicates
the IP address of the web server serving
the request.
IP/FQDN are both valid.
<web>
<client>203.122.33.1</client>
<server>www.indwide.net</server>
</web>
-->
</source>
 <!--

DELIVERY DETAILS
Valid Types: address, pos (point of sale)
If delivery details are absent, and source is web
they will try to be looked up via the sessionid
If delivery details are <pos>, it is assumed the
goods were taken from point of sale.
<delivery>
<address>
<unit></unit> or <level></level> are optional
<number></number>
<street></street>
<suburb></suburb>
<city></city>
<state></state>
<country></country>
<postcode></postcode>
</address>
</delivery>
or
<delivery>
<pos/>
</delivery>
-->
 <delivery>
<pos />
</delivery>
```

```
 <!--
PAYMENT DETAILS (SENSITIVE)
Valid types: card, iwnuser
Note the iwnuser id may be looked up using the
card number for other projects such as loyatly.
-->
 <payment>
 <card>
<name>Card N. Holder</name>
<number>456400000000</number>
 <!--
month can be the following:
01, 02, ..., 11, 12
1, 2, ..., 11, 12
January, February, ..., November, December
Jan, Feb, ..., Nov, Dec
year can be the following:
00, 01, ..., 98, 99 (+2000)
2000, 2001, ..., 2998, 2999
-->
<expiry month="01" year="01" />
 <!--
valid account types are credit.
cheque, savings, etc. will be supported
in a later release.
-->
<account>credit</account>
</card>
 <!--
Alternately, <iwnuser> can be used to lookup the
credit details via the IWNUserID or email
<iwnuser>
<id>3</id>
<password>foo</password>
</iwnuser>
or
<iwnuser>
<email>foo@bar.com</email>
<password>foo</password>
</iwnuser>
-->
</payment>
 <!--
PRODUCTS
Optional. The <products> collection may be ommitted entirely
if desired.
-->
 <products>
 <!--
Each product must contain a product id and
a merchant id. This allows for multiple products/merchants
per purchase request (example: Virtual Mall)
-->
 <product merchantid="99">
<productid type="EAN">1111111111111</productid>
 <!--
name and description are optional
-->
<name>foo</name>
<description>600gram bar of foo</description>
 <!--
unit and unit name refer to the quantity
of the product.
examples:
1 x 600ml milk
<id>1</id>
<description>milk</description>
<unit name="ml">600</unit>
<quantity number="1">...</quantity>
3 x 3kg bag of oranges
<id>2</id>
<description>3kg bag of oranges</description>
```

```
<unit name="kg">3</unit>
<quantity number="3">...</quantity>
-->
<unit name="kg">1</unit>
 <!--

currency codes conform to ISO 4217

-->
 <quantity number="2">
 <!--
cost tags are optional within <quantity/> tags.
There may be an arbitrary amount of cost tags here.
They are not used for calculation of price at all.
They are solely used for reporting/accounting.
It is up to the merchants discretion which costs
they send with each purchase request.
Cost names are arbitrary.
Costs within the <quantity> tag refer to each unit,
not to the group of products.
-->
<cost name="cost" currency="AUD">100</cost>
<cost name="pretax" currency="AUD">200</cost>
<cost name="sale" currency="AUD">220</cost>
<cost name="staffdiscount" currency="AUD" rate="10%">20</cost>
<cost name="dth-fee" currency="AUD" rate="10%">20</cost>
<cost name="GST" currency="AUD" rate="10%">16</cost>
<cost name="subtotal" currency="AUD">176</cost>
</quantity>
 <!--
There may be an arbitrary number of <cost> tags within each
<product> tag.
For a description, see above.
Cost tags here are also optional.
Cost tags inside <product> refer to the pricing of the entire
<quantity> of products.
-->
<cost name="foo" currency="AUD">10</cost>
<cost name="quantitytotal" currency="AUD">362</cost>
</product>
</products>
 <!--
COST

Cost outside the <products> tree is mandatory.
This value is the actual value that will be passed
to the financial switch for credit/debit.

-->
<cost name="total" currency="AUD">362</cost>
</request>
</purchase>
</iwn>
```

Purchase Response

```
<?xml version="1.0" ?>
<!DOCTYPE iwn (View Source for full doctype...)>
 <!--
IWN Purchase Response Sample Document
Version: SId: purchase-response.xml,v 1.4 2000/06/05 03:05:08 dth Exp $
Milestone: 4

Version corresponds to the Milestone release.

id corresponds to the Purchase ID.
-->
 <iwn version="0.4" session="6d518aa77a49d8bc823037111dd877d6931d6245">
 <purchase id="acs233d23dacad">
 <response>
<code type="purchase">0</code>
<description>Purchase Successful</description>
<rrn>iwn0032114125511234</rrn>
 <!--

- Request Transaction Status
- Request Purchase Authorization
- Request Purchase Reversal
- Logout

-->
 <options>
<workflow id="2">Logout</workflow>
<workflow id="5">Purchase Authorize</workflow>
<workflow id="6">Purchase Reversal</workflow>
<workflow id="7">Transaction Status</workflow>
</options>
</response>
</purchase>
</iwn>
```

IWN Transaction SDK

API User Guide

Version	1.6
Date	27 May 2000

Contents

Preface

This document is provided to assist a developer in utilising the IWN Transaction SDK in the development of a client application that is to perform transactions using the IWN Transaction System.

Associated Documentation

This document can be used in conjunction with the *IWN Engine Client Specifications*.

Overview

The IWN Transaction SDK consists of an ActiveX EXE containing several classes that are used to collect, format, verify and send transactions to the IWN Transaction System, and receive, parse and provide the results of those transactions to a calling application.

Communications

The IWN Transaction Classes communicate with the IWN Server over TCP/IP using specially formatted documents for transferring information. Each Transaction type may perform any number of Transactions with the IWN Server. For a Transaction the flow of information is as follows:

- A TCP/IP Connection is established between the Client and the IWN Server.
- The Client sends a Request Document to the IWN Server.
- The IWN Server sends a Response Document to the Client.

Sessions

The IWN Transaction Systems utilises Session management to provide tracking, reconciliation and to ensure the integrity of Transactions.

When a Client creates an instance of the Session Object, it can call the Logon method which will request a Session from the IWN Server. Once the IWN Server has issued the Client with a valid Session, the Client can use that Session to send Transactions.

A Client uses the Merchant Details provided to identify and authorize itself to the IWN Server.

Transactions

The following Transaction types are currently supported: PURCHASE, REVERSAL, AUTHORIZE and STATUS

Sample Session

Here is a typical usage scenario.

This section refers to code in the iwnExample Visual Basic project distributed with the iwn component.

- The calling application will typically create an IWN session object when it loads or initialises.
- It will then set the following Session object properties:
    - ServerAddress
    - ServerPort
- The calling application may then ensure it has contact with the IWN engine and call the Session objects Login method, passing the Merchants Username and password.
- The Login method should return true or false and set an error depending on whether the Login was successful. The application may wish to rectify any problems and try the login again should the login have failed.
- It the Login method returns True, the the Merchant has been issues a valid session and may now commence with committing transactions.
- To perform a transaction, the calling application would execute the Session objects Transaction collections Add method, which will return a valid transaction object. When the Add method is called, the IWN transaction server is contacted for a valid transaction number to use. If this failed, then the Add method will return null. The calling application should check the validity of the Transaction object returned by the Add method before it attempts to use it.
- When it has obtained a valid transaction object, the calling application must then set the following properties of that transaction, before the transaction can be committed:
    - CardExpiryMonth
    - CardExpiryYear
    - CardName
    - CardNumber
    - CurrencyType
- The calling application can then create Product objects within the Transaction object to represent the difference products, prices and quantities for the transaction. The calling

application can add a Product object to the Transaction by calling the Transactions objects Products collection Add method. The Add method will return Product object, whose properties can be set to indicate the desired product, quantity and unit cost.

- The calling application can repeat this step to add any number of products to the transaction before it is commited.
- Before the Transaction is committed, the calling application can adjust the amount for the transaction by setting the Transaction objects Amount property. This allows for adjustments such as discounts, part payments or credits. If there are no products added, the calling application must explicitly set the Transaction objects Amount property or the Transaction will have the default value of 0.
- When the calling application wishes to commit the transaction, it can simple call the Transaction objects Send method, which will send the transaction to the IWN engine, and return True if successful or False if the transaction failed.

Terminology

IWN Server

The IWN Transaction Server.

Client

The calling application, in which objects of the IWN Transaction Classes are created.

Transaction

A single **network** transaction. Indication the sending of data to the IWN Server and the receipt of information from the IWN Server. This **DOES NOT** indicate a financial transaction. A transaction is considered complete when a valid Request Document has been sent to the IWN Server and a valid Response Document has been received from the IWN Server

Request Document

The IWN Transaction Classes use a special message format for transferring information between a Client and the IWN Server. As in a Transaction, a document is sent and a document is received. A document transferred **FROM** the Client **TO** the IWN Server is a Request Document.

Response Document

As with the Request Document, the Response Document is sent **FROM** the IWN Server **TO** the client after the server has received and processed a valid Request Document.

Purchase

A Purchase is a type of credit transaction. Within the scope of the IWN Transaction SDK it is considered a virtual transaction. A Purchase may constitute several Transactions, depending on the requirements of the Purchase. *(See **Transaction**)*

Reversal

A Reversal is a type of credit transaction. *(See **Purchase**)*

Session

Objects

Session Object

Parent: (none)

Children: Transactions

Default Property: (none)

Properties

Name	Type	Access
MerchantID	String	Read/Write
Route	String	Read/Write
ServerAddress	String	Read/Write
ServerPort	Long	Read/Write
SessionID	String	Read/Write
TerminalID	String	Read/Write
Transactions	Transactions Collection Object	Read Only

Methods

Name	Parameters
Login	*Email* as String *Password* as String
Logout	(none)

Comments

A single session object is created, it's parameters are set and the login method is called. Once a successful login has occurred the session object can be used to create transaction objects. Transaction objects cannot (read should not) be created if there is no current valid session, or if the session object is not connected to a valid session.

Transactions Collection Object

Properties

Name	Type	Access
Count	String	Read Only
Item	Object	Read Only

Methods

Name	Parameters
Add	(none)
Delete	*Index* as Variant

Comments

The Transactions Collection object exists within the session object and holds all the Transaction objects that have been created within the session. The Transactions Collection object can be enumerated in For Each statements to retrieve each Transaction object in sequence.

Transaction Object

Properties

Name	Type	Access
Amount	Long	Read/Write
CardExpiryMonth	Integer	Read/Write
CardExpiryYear	Integer	Read/Write
CardName	String	Read/Write
CardNumber	String	Read/Write
Code	Integer	Read Only
CurrencyType	*iwnCurrencyType* Integer	Read/Write
Description	String	Read Only
Products	Products Collection Object	Read Only
RRN	String	Read/Write
SessionID	String	Read Only
SystemTrace	Integer	Read Only
TransactionID	String	Read Only

Methods

Name	Parameters
Send	(none)

Comments

A Transaction Object is created within the Transactions Collection object when there is a valid session present in the Session object. A Transaction object is created, its properties set, and the send method is called. Transactions will exist until explicitly killed or until as session has ended.

Products Collection Object

Properties

Name	Type	Access
Count	String	Read Only
Item	Object	Read Only

Methods

Name	Parameters
Add	(none)
Delete	*Index* as Variant

Comments

The Products Collection object exists within a Transaction object and holds all the Product objects that have been created within the transaction. The Products Collection object can be enumerated in For Each statements to retrieve each Product object in sequence.

Product Object

Properties

Name	Type	Access
Amount	Long	Read/Write
CurrencyType	*iwnCurrencyType* Integer	Read/Write
MerchantID	String	Read/Write
ProductID	String	Read/Write
Quantity	Long	Read/Write

Methods

Comments

A Product Object is created within the Products Collection object in a valid Transaction Object. A Product object is created, its properties set. The product details are issued with the Transaction when the parent Transaction object is committed.

Products will exist until explicitly killed or until the parent Transaction Object has terminated has ended.

Properties

MerchantID Property (Session Object, Transaction Object, Product Object)

The MerchantID property contains the Merchant ID for the Session, Transaction or Product.

Syntax

objSession.**MerchantID**

Data Type

String

Comments

The MerchantID property can be set before an active session. Unless explicitly set, MerchantID's for Transaction and Product objects will assume the value of MerchantID in the Session object.

Example

With objSession

.MerchantID = "99"

End With

Route Property (Session Object)

The Route property is used to specify a gateway IP address to use when sending transactions.

Syntax

objSession.**Route**

Data Type

String

Comments

Setting the Route property will modify the system routing table by adding a static route to the IP address or hostname specified by the ServerAddress property via the IP address specified in the Route property. The Route property can be set any number of times during the life of an active session and the next Transaction to be commited will use the route specified.

If the IP address supplied with the Route property does not exist as a valid interface, the routing table will not be modified but the Route property will retain the IP address given. Currently there is no way to determine if a call to the routing table was successful, but future versions will support notification of the Route entry.

Example

This code will add a route to the system routing table specifying that all traffic for iwnengine.indwide.net should go through the interface 192.168.0.10

```
With objSession
    .ServerAddress = "iwnengine.indwide.net"
    .Route = "192.168.0.10"
End With
```

ServerAddress Property (Session Object)

The ServerAddress property specifies the address of the IWN engine server that the Session should communicate with.

Syntax

objSession.**ServerAddress**

Data Type

String

Comments

The ServerAddress property is required before the Session objects Login method can be called. The ServerAddress property can (read should) only be set once before the session is validated. Setting this property during a valid session will not (read should not) take effect until the Session is terminated and a new Session is created.

Example

With objSession

.ServerAddress = "iwnengine.indwide.net"

.ServerPort = 5005

End With

ServerPort Property (Session Object)

Sets the TCP port used to communicate with the IWN engine server specified with the ServerAddress property.

Syntax

objSession.**ServerPort**

Data Type

String

Comments

See ServerAddress Property (Session Object)

Example

With objSession

.ServerAddress = "iwnengine.indwide.net"

.ServerPort = 5005

End With

See Also

ServerAddress Property (Session Object)

SessionID Property (Session Object)

The SessionID property containts the Session ID for the active session.

Syntax

objSession.**SessionID**

Data Type

String

Comments

The SessionID property is automatically assigned a value when a session is created by executing the Login method. Functionallty has been provided to set the SessionID property of a Session object to support the future ability to resume sessions between invocations of the Session object. This functionality is not currently available and setting the SessionID before or after a session login will not (read should not) affect any transactions that are committed within the scope of the session.

Example

TerminalID Property (Session Object)

The TerminalID property contains the TerminalID for the session.

Syntax

objSession.**TerminalID**

Data Type

String

Comments

The TerminalID is required before a valid session can be obtained.

Example

With objSession

.TerminalD = "TI001"

End With

Transactions Property (Session Object)

The Transactions Property returns a single Transaction object or a Transactions collection object.

Syntax

Set *objTransactions* = *objSession*.**Transactions**

Set *objTransaction* = *objSession*.**Transactions**(*index*)

Set *objTransaction* = *objSession*.**Transactions**(*name*)

objSession

Object. A Session object.

objTransactions

Object. A Transactions collection object.

objTransaction

Object. A Transaction Object.

index

Long. Specifies the number of the Transaction within the Transactions collection. Ranges from 1 to the value specified by the Transactions collection's Count Propery.

name

String. A Transaction ID.

Data Type

Object (Transaction or Transactions Collection)

Comments

Example

Amount Property (Transaction Object)

The Amount property contains the amount for the transaction.

Syntax

objSession.**Amount**

Data Type

Long

Comments

The amount property is represented in base units of currency. Example, for $AUD, the long contained in the Amount property would represent cents.

The amount property represents the amount for the transaction. This must be set by the application if the amount for the transaction is different than the amount calculated as the total of the associated product objects. The Amount property will (read should) default to the total of the associated product objects when the transaction is commited.

Example

```
With objSession
    .Amount = 133414
End With
```

CardExpiryMonth Property (Transaction Object)

The CardExpiryMonth property represents the 2 digit expiry month of the credit card specified by the CardNumber property.

Syntax

objSession.**CardExpiryMonth**

Data Type

Integer

Comment

The CardExpiryMonth Property accepts and Integer in the range 1 – 12 representing the 2 digit month of the expiry date on a credit card. If the value given when setting the CardExpiryMonth property is outside this range, then the property defaults back to 0 and the transaction will return an error indicating an invalid expiry date when an attempt is made to commit the transaction.

Example

With objSession

.CardExpiryMonth = 4

.CardExpiryYear = 0

End With

See Also

CardExpiryYear Property

CardExpiryYear Property (Transaction Object)

The CardExpiryYear property represents the 2 digit expiry year of the credit card specified by the CardNumber property.

Syntax

objSession.**CardExpiryYear**

Data Type

Integer

Comments

The CardExpiryYear Property accepts a year value as an integer in both 2 and 4 digit format. If the specified Integer is between 0 and 99 it is assumed that this represents a 2 digit date between the year 2000 and 2099. If the specified Integer is between 2000 and 2099 it is assumed that this represents a 4 digit date between the year 2000 and the year 2099.

See Also CardExpiryMonth Property for handling of values outside the accepted ranges.

Example

See CardExpiryMonth Property.

See Also

CardExpiryMonth Property.

CardName Property (Transaction Object)

The CardName property contains the full name of the card holder of the credit card specified in the CardNumber property.

Syntax

objSession.**CardName**

Data Type

String

Comments

Example

See Also

CardNumber Property (Transaction Object)

CardNumber Property (Transaction Object)

The **CardNumber** property contains the card number of the credit card that is referenced by the transaction.

Syntax

objTransaction.**CardNumber**

Data Type

String

Comments

The **CardNumber** property is required before a transaction can be committed.

Example

This example represents setting the credit card details for a transaction object. The card details shown here would be from a card belonging to John Citizen with a number of 4502 0000 0000 0000 that expires on January 2000.

```
With objTransaction
    .CardNumber = "4502000000000000"
    .CardName = "John Citizen"
    .CardExpiryMonth = 1
    .CardExpiryYear = 0
End With
```

See Also

CardName Property (Transaction Object)

CardExpiryMonth Property (Transaction Object)

CardExpiryYear Property (Transaction Object)

Code Property (Transaction Object)

The **Code** property returns the response code received from the IWN Engine after a Transaction has been committed.

Access

Read Only

Syntax

objTransaction.**Code**

Data Type

Integer

Comments

Example

See Also

IWN Engine Error and Return Codes

CurrencyType Property (Transaction Object)

The **CurrencyType** property contains the the Currency type of the amount specified in the **Amount** property.

Access

Read/Write.

Syntax

ObjTransaction.**CurrencyType**

Data Type

iwnCurrencyType

Comments

The value contained in the **CurrencyType** property represents a value from the **iwnCurrencyType** enumerator.

Example

The following example would set the result in a Transaction for $100.00 Australia Dollars.

```
With objTransaction
    .CurrencyType = AUD
    .Amount = 10000
End With
```

See Also

Amount Property (Transaction Object)

Description Property (Transaction Object)

The **Description** property returns the description property received from the IWN engine after a transaction has been committed.

Access

Read Only.

Syntax

objTransaction.**Description**

Data Type

String.

Comments

Example

Dim sDescription As String

sDescription = objTransaction.Description

See Also

Code Property (Transaction Object)

RRN Property (Transaction Object)

Products Property (Transaction Object)

The **Products** property returns a single Product object or a Products collection object.

Access

Read Only.

Syntax

Set *objProductss* = *objTransaction*.**Products**

Set *objProduct* = *objTransaction*.**Products**(*index*)

Set *objProduct* = *objTransaction*.**Products**(*name*)

objTransaction

Object. A Transaction object.

objProducts

Object. A Products collection object.

objProduct

Object. A ProductObject.

index

Long. Specifies the number of the Product within the Products collection. Ranges from 1 to the value specified by the Products s collection's Count Property.

name

String. A Product ID. This can be identified as the Product objects **ProductID** property.

Data Type

Object (Product or Products collection)

Comments

Example

The following example would retrieve the Product object for a product with a product id of 131001 from the Products collection in the Transaction object.

```
Dim objProduct As Product

Dim objTransaction As Transaction

Set objProduct = objTransaction.Products("131001")
```

The following example would iterate through all the products in the Transaction objects Products collection and display the Product objects ***ProductID*** *property.*

```
Dim objTransaction As Transaction

Dim iCount As Integer

For iCount = 1 To objTransaction.Products.Count – 1

    Debug.Print objTransaction.Products(iCount).ProductID

Next
```

See Also

Product object.

RRN Property (Transaction Object)

The **RRN** property returns the receipt number of the response received from the IWN engine after a transaction has been committed.

Access

Read Only.

Syntax

objTransaction.**RRN**

Data Type

String.

Comments

Example

Dim sRRN As String

sRRN = objTransaction.RRN

See Also

Code Property (Transaction Object)

Description Property (Transaction Object)

SessionID Property (Transaction Object)

The **SessionID** property returns the Session ID of the parent session under which the Transaction was created.

Access

Read Only.

Syntax

objTransaction.**SessionID**

Date Type

String

Comments

Example

See iwnexample example.

See Also

Session object

SessionID Property (Session Object)

SystemTrace Property (Transaction Object)

The **SystemTrace** property returns the system trace code of the response received from the IWN engine after a transaction has been committed.

Access

Read Only.

Syntax

objTransaction.**SystemTrace**

Date Type

Integer.

Comments

Example

Dim iSystemTrace As Integer

iSystemTrace = objTransaction.SystemTrace

See Also

Code Property (Transaction Object)

Description Property (Transaction Object)

RRN Property (Transaction Object)

TransactionID Property (Transaction Object)

The **TransactionID** property returns the transaction id of the transaction object.

Access

Read Only.

Syntax

objTransaction.**TransactionID**

Date Type

String.

Comments

The **TransactionID** property is automatically assigned by the IWN engine through the Session object when the Transaction object is created. After the transaction has been created, the transaction id cannot be changed.

Example

See Also

Count Property (Transactions Object)

Returns the number of items in the **Transactions** collection.

Access

Read Only.

Syntax

objTransactions.**Count**

Date Type

Long.

Comments

Example

See Also

Count Property (Products Object)

Returns the number of items in the **Products** collection.

Access

Read Only.

Syntax

objProducts.**Count**

Date Type

Long.

Comments

Example

See Also

Item Property (Transactions Object)

Returns a **Transaction** object from a **Transactions** collection. The **Item** property is the default property of the **Transactions** object.

Access

Read Only

Syntax

objTransactions.**Item**(*index*)

objTransactions.**Item**(*key*)

index

Long. A number in the range of 1 to the value of the **Transactions** object **Count** property.

key

String. The transactions id of an individual **Transaction** object.

Date Type

Object. **Transaction** object.

Comments

Example

See Also

Item Property (Products Object)

Returns a **Product** object from a **Products** collection. The **Item** property is the default property of the **Products** object.

Access

Read Only

Syntax

objProducts.**Item**(*index*)

objProducts.**Item**(*key*)

index

Long. A number in the range of 1 to the value of the **Products** object **Count** property.

key

String. The product id from an individual **Product** object.

Date Type

Object. **Product** object.

Comments

Example

See Also

Methods

Login Method (Session Object)

The **Login** method contacts the IWN engine and requests a session in which to transfer transactions.

Syntax

objSession.**Login**(*Email, Password*)

objSession

Required. Object. Session Object.

Email

Required. String. Email address, also referred to as username.

Password

Required. String. Password associated with the supplied username.

Return Type

Boolean

Comments

The **Login** method contacts the IWN engine, requests a session and retrieves a session id. If the **Login** method is successful, the **Login** method will return **True**. If the **Login** method cannot obtain a session id for any reason it will return **False** and set an error.

Example

See Also

Logout Method (Session Object)

SessionID Property (Session Object)

Logout Method (Session Object)

The **Logout** method closes a session created by the **Login** method and clears any stored session information.

Syntax

objSession.**Logout**

Return Type

Boolean

Comments

The **Logout** method will close a open session and clears all session related data.

Example

See Also

Login Method (Session Object)

Add Method (Transactions Collection Object)

Adds a **Transaction** object to a transactions collection.

Syntax

Set *objTransaction* = *objTransactions*.**Add**

objTransaction

If the **Add** method returns **True**, contains the new **Transaction** object.

objTransactions

Required. The **Transactions** collection object.

Return Type

Object. **Transaction** object.

Comments

The **Add** method contacts the IWN server for a valid transaction id. If the **Add** method can obtain a valid transaction id, then it will return a valid **Transaction** object.

Example

See Also

Delete Method (Transactions Collection Object)

This method has been disabled and is under review. It should not be utilised in a calling application.

Deletes a **Transaction** object from a **Transactions** collection object.

Syntax

Return Type

Comments

Example

See Also

Send Method (Transaction Object)

Commits a transaction to the IWN engine.

Syntax

objTransaction.**Send**

Return Type

Boolean.

Comments

The **Send** method queues a transaction for sending to the IWN engine. If the **Send** method was able to successfully que the transaction, it will return **True**. If the **Send** method is unable to queue a transaction, it will return **False** and set an error. The Send method is asynchronous, a return value of **True** does not indicate that the transaction was sent successfully, it only means that the transaction was successfully queued. When a transaction is sent, the **Send** method will raise the **Response** event in its parent **Session** object, indicating whether the transaction was successfully sent. If the **Send** method successfully transmits the transaction to the IWN engine and receives a valid response, it will set the values of the **Code**, **Description** and **SystemTrace** properties to those received in the response.

Example

See Also

Add Method (Products Collection Object)

Adds a **Product** object to a products collection.

Syntax

Set *objProduct* = *objProducts*.**Add**

objProduct

If the **Add** method returns **True**, contains the new **Products** object.

objProducts

Required. The **Products** collection object.

Return Type

Object. **Product** object.

Comments

Example

See Also

Delete Method (Products Collection Object)

This method has been disabled and is under review. It should not be utilised in a calling application.

Deletes a **Product** object from a **Products** collection object.

Syntax

Return Type

Comments

Example

See Also

Events

Enumerations

Error Codes

Code	Number	Description
ERR_SUCCESS	0	Success.
ERR_TRANSACTION	1000	There was a general transaction error.
ERR_INCOMPLETE_REQUEST	1001	
ERR_INVALID_REQUEST	1002	
ERR_INVALID_RESPONSE	1008	
ERR_REPLY_TIMEOUT	1009	
ERR_SEND_FAILED	1010	
ERR_IN_PROGRESS	1900	
ERR_SOCKET	2000	
ERR_SOCKET_CONNECT_TIMEOUT	2001	
ERR_SOCKET_NOT_READY	2002	
ERR_INVALID_SERVER_DETAILS	2003	
ERR_SOCKET_EVENT_UNKNOWN	2100	
ERR_SYSTEM	7000	

Claims

1. An electronic commerce system comprising:

a series of point of sale terminals providing for point of sale information handling of a number of businesses:

an interconnection network interconnecting the point of sale terminals to a central database facility:

a central database facility for storing information about each of said businesses for access by the operators of said point of sale terminals: and

a series of service providers interconnected to said central database facility for meeting requests issued by said point of sale terminals.

2. A system as claimed in claim 1 further comprising:

a series of suppliers interconnected to said central database facility for meeting requests issued by said point of sale terminals.

3. system as claimed in claim 2 wherein said suppliers include at least one of:

an import/expert agent; a warehousing agent or a producer.

4. A system as claimed in any previous claim wherein said service providers include one of:

a third party information vendor providing information upon request;

a financial transaction vendor providing financial transaction authorisation upon request; or

an order fulfilment vendor providing order fulfilment upon request.

5. A system as claimed in any previous claim wherein said point of sale terminals include local database information and programs which are downloaded on demand from said central database facility.

6. A system as claimed in any previous claim wherein the point of sale terminals to interact with the said main remote data-store wherein any changes to the local datastore are periodically updated to the main datastore, and where relevant produced as part of the clients web page.

7. A system as claimed in any previous claim further comprising access means for accessing the datastore as a member of the general public using a web browser and further means for communicating in a timely manner directly to the Point of Sale merchant and if that merchant is there then communicating with the merchant in actual time.

8. A system as claimed in any previous claim wherein said request are tranmitted in the form of XML documents or the like to and from said central database facility.

9. A system as claimed in any previous claim further comprising a series of user mobile data entry devices which interact with said point of sale terminals in the authorization of a transaction.

10. A system as claimed in claim 9 wherein said mobile data entry device include one of WAP enabled phones, mobile phones or bluetooth connected devices.

11. A system as claimed in any previous claim further comprising a separate interaction unit for users to interact with said central database facility for the viewing of transaction statistics associated with said system.

12. A system as claimed in claim 11 wherein said viewing of transaction statistics includes utilising OLAP facilities on said central database.

13. A system as claimed in any previous claim wherein requests are provided by a software development kit applications programming interface.

14. A system as claimed in any previous claim wherein actions undertaken by said database facility are in the form of workflow steps executed by the facility.

15. A system as claimed in claim 14 wherein said workflow is spawned by the template structure of said request.

16. A system as claimed in any previous claim further comprising an interactive graphical database for interacting with said central database facility.

17. A system as claimed in any previous claim wherein multiple centralised database facilities are provided and interact with one another to perform functions.

18. An electronic commerce system substantially as herein before described with reference to the accompanying drawings.

19. A method of conducting commercial transactions substatantially as hereinbefore described with reference to Fig. 3 of the drawings.

20. A method of conducting electronic commerce substantially as hereinbefore described with reference to the accompanying drawings.

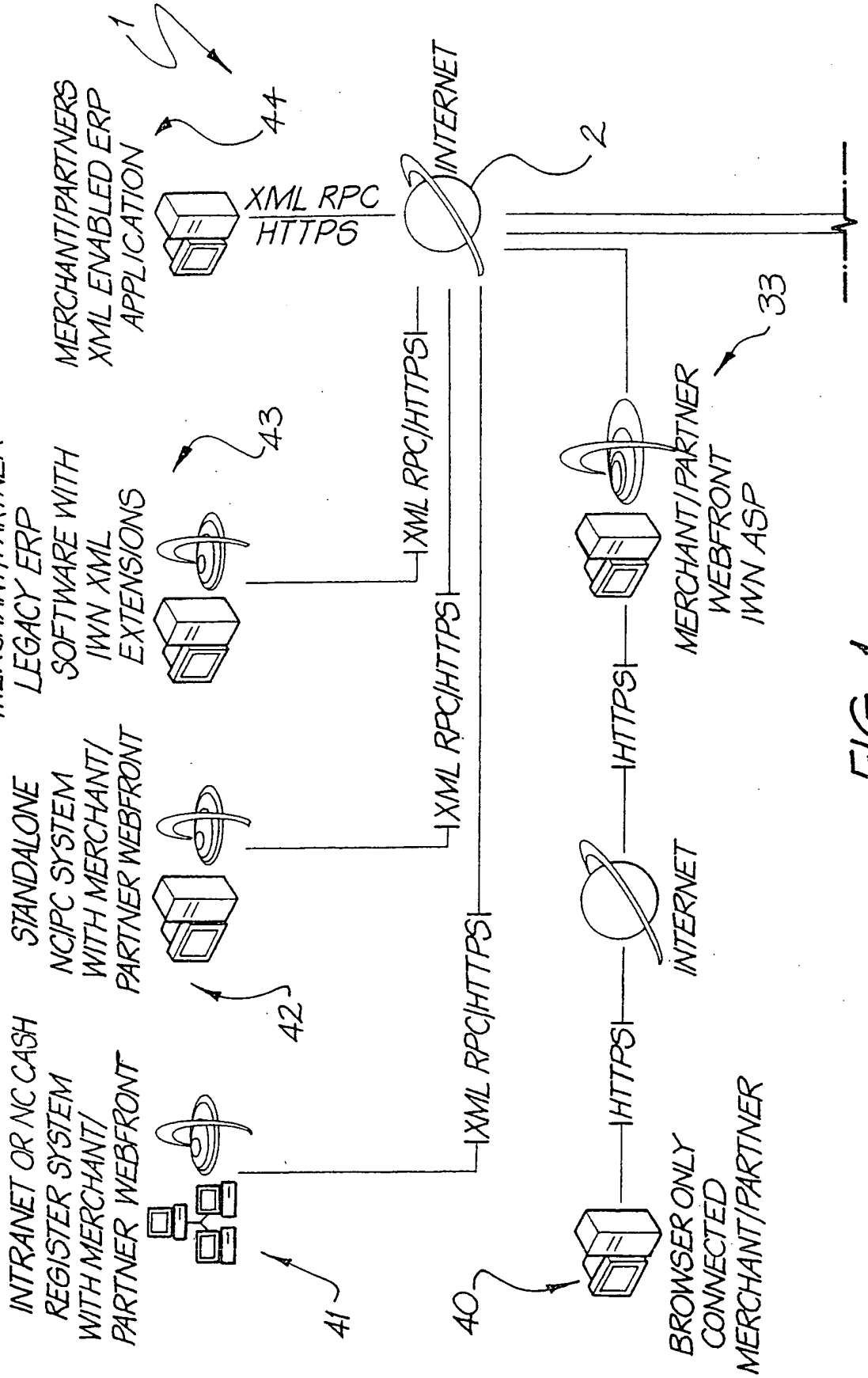

INTRANET OR NC CASH REGISTER SYSTEM WITH MERCHANT/ PARTNER WEBFRONT
STANDALONE NC/PC SYSTEM WITH MERCHANT/ PARTNER WEBFRONT
MERCHANT/PARTNER LEGACY ERP SOFTWARE WITH IWN XML EXTENSIONS
MERCHANT/PARTNERS XML ENABLED ERP APPLICATION
XML RPC HTTPS
INTERNET
XML RPC/HTTPS
XML RPC/HTTPS
XML RPC/HTTPS
HTTPS
HTTPS
INTERNET
MERCHANT/PARTNER WEBFRONT IWN ASP
BROWSER ONLY CONNECTED MERCHANT/PARTNER
41
42
43
44
1
2
33
40

FIG. 1

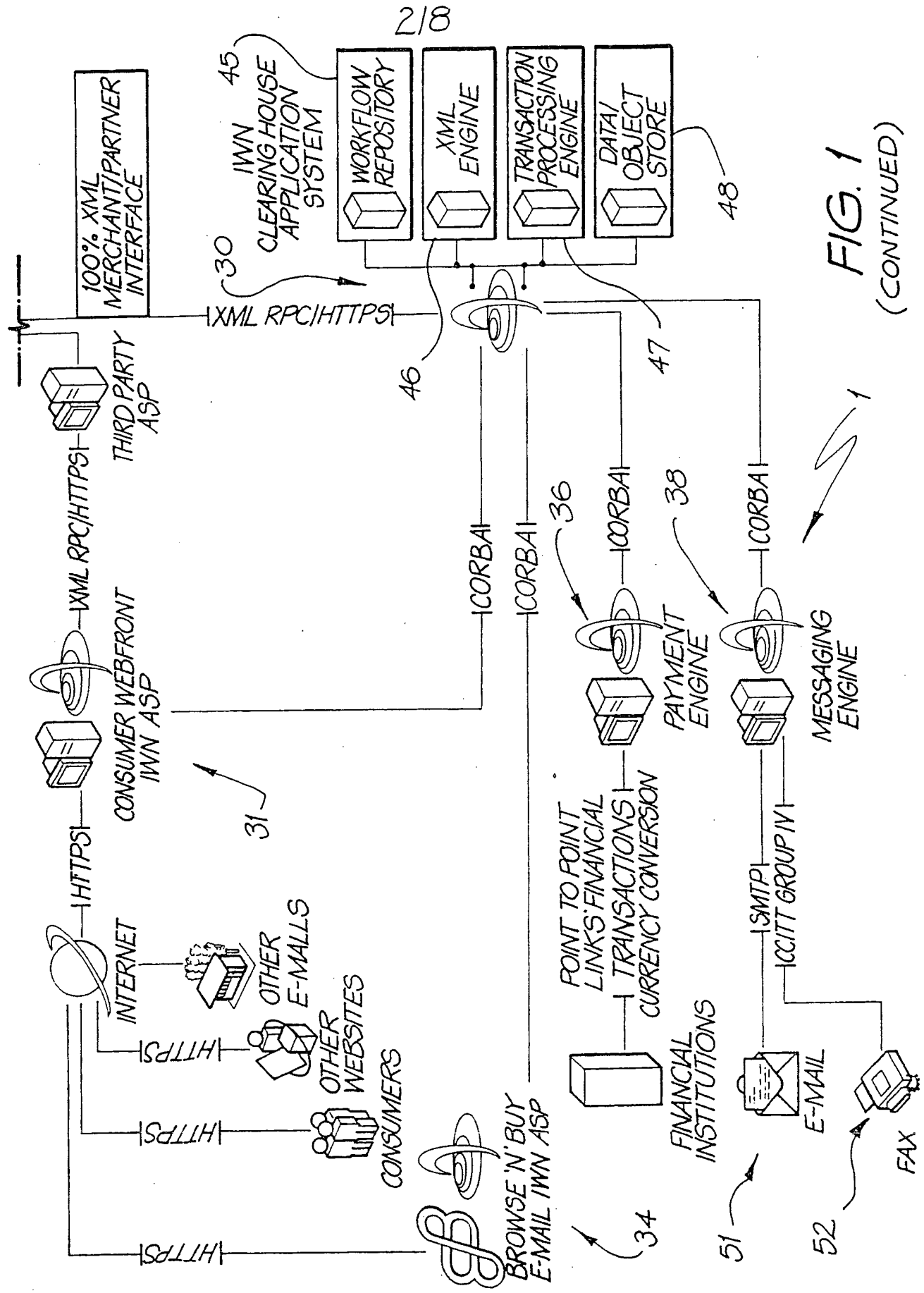


FIG. 1 (CONTINUED)

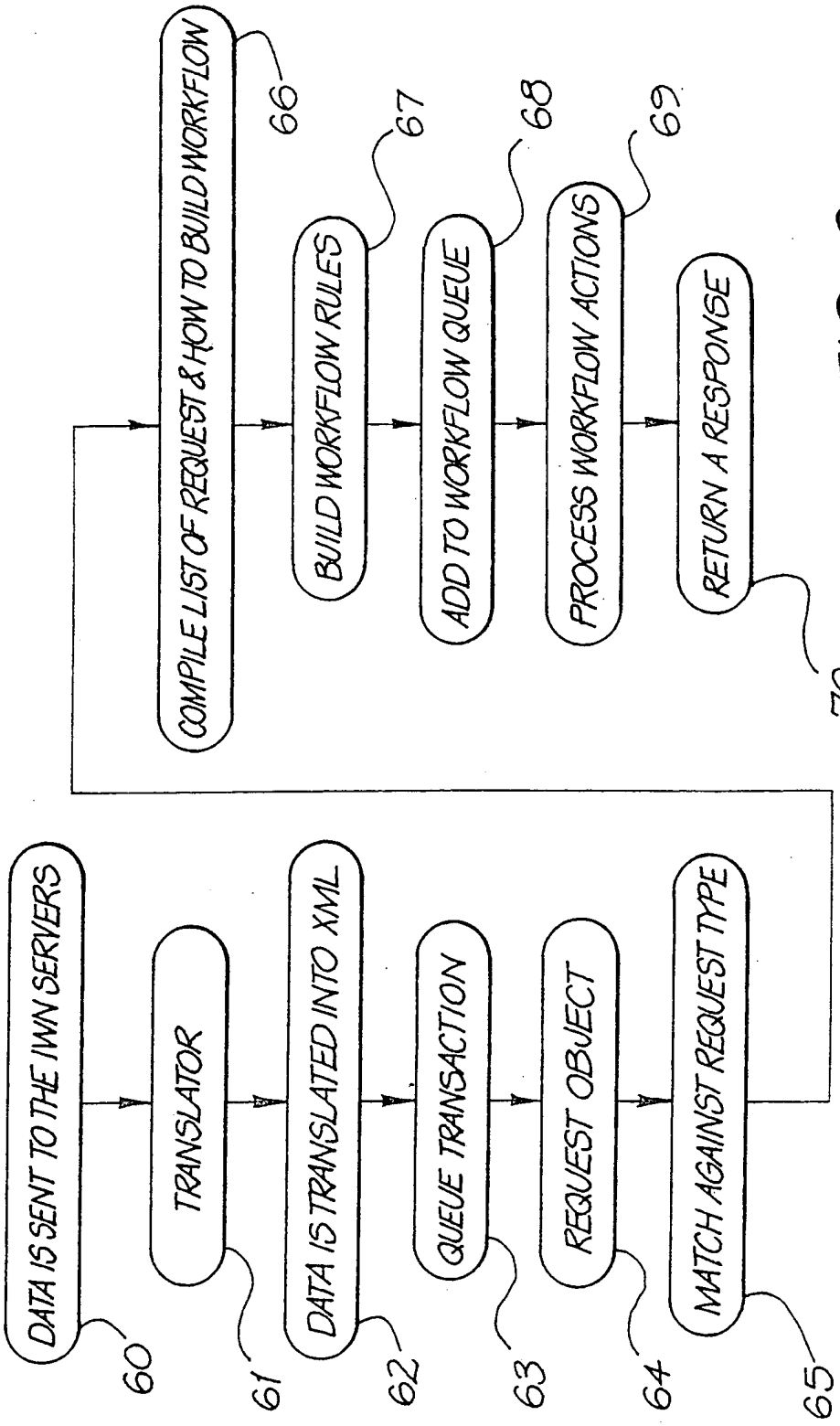

DATA IS SENT TO THE IWN SERVERS
60
TRANSLATOR
61
DATA IS TRANSLATED INTO XML
62
QUEUE TRANSACTION
63
REQUEST OBJECT
64
MATCH AGAINST REQUEST TYPE
65
COMPILE LIST OF REQUEST & HOW TO BUILD WORKFLOW
66
BUILD WORKFLOW RULES
67
ADD TO WORKFLOW QUEUE
68
PROCESS WORKFLOW ACTIONS
69
RETURN A RESPONSE
70

FIG. 2

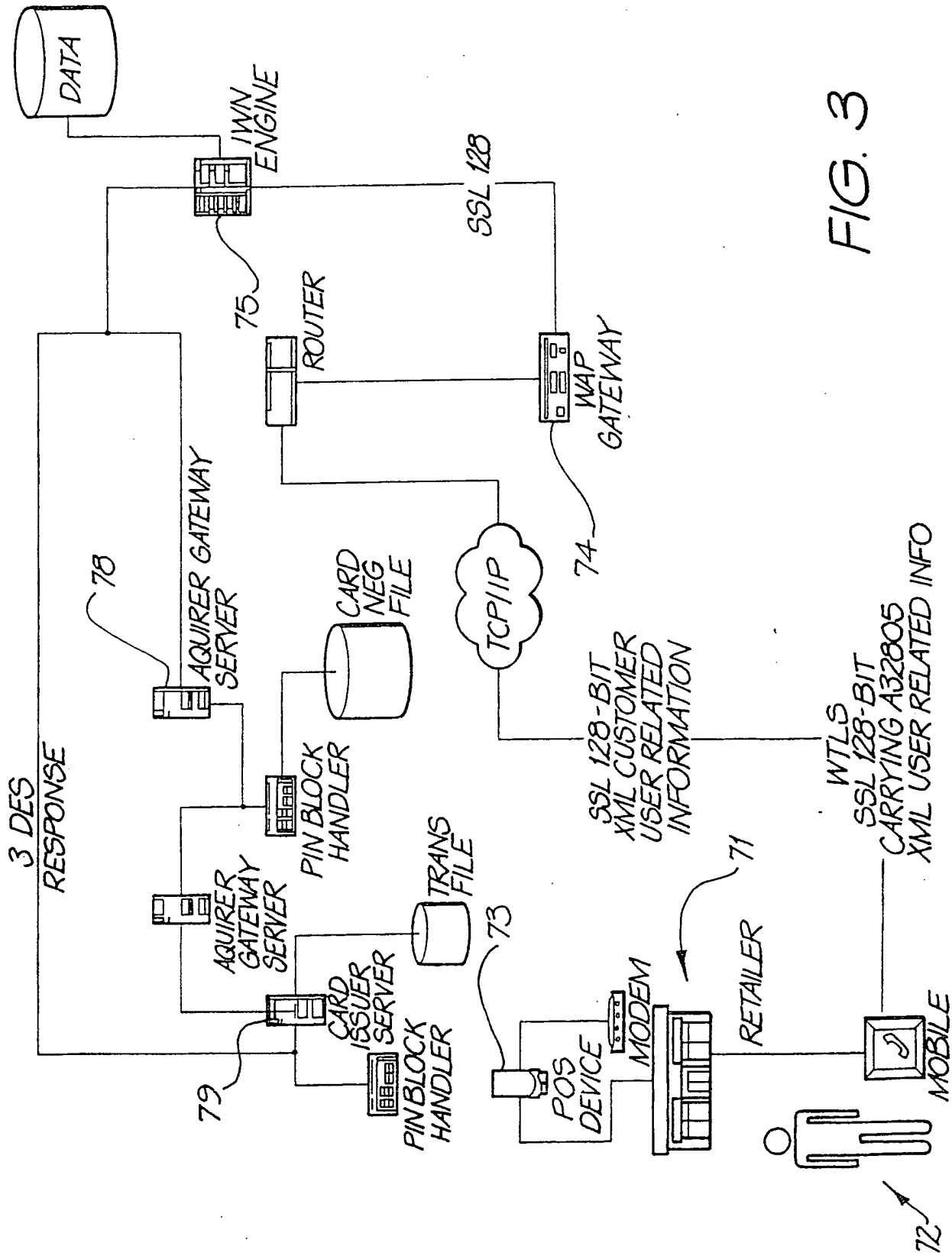

DATA
IWN ENGINE
SSL 128
75
ROUTER
WAP GATEWAY
74
78
AQUIRER GATEWAY SERVER
CARD NEG FILE
TCP/IP
3 DES RESPONSE
PIN BLOCK HANDLER
SSL 128-BIT XML CUSTOMER USER RELATED INFORMATION
WTLS SSL 128-BIT CARRYING A32805 XML USER RELATED INFO
AQUIRER GATEWAY SERVER
TRANS FILE
71
73
CARD ISSUER SERVER
PIN BLOCK HANDLER
POS DEVICE
MODEM
RETAILER
MOBILE
79
72

FIG. 3

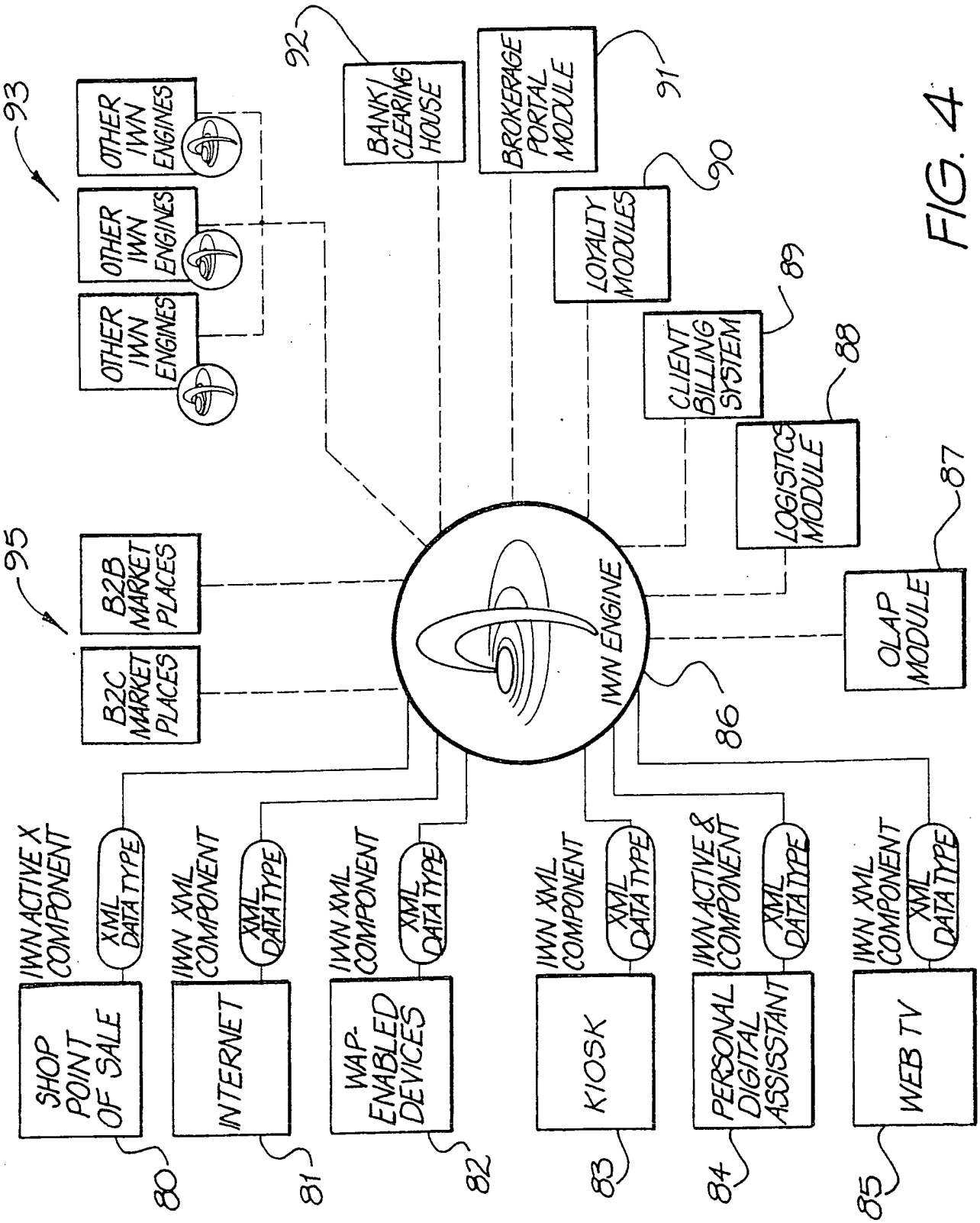

IWN ACTIVE X COMPONENT
XML DATA TYPE
SHOP POINT OF SALE
80
IWN XML COMPONENT
XML DATA TYPE
INTERNET
81
IWN XML COMPONENT
XML DATA TYPE
WAP-ENABLED DEVICES
82
IWN XML COMPONENT
XML DATA TYPE
KIOSK
83
IWN ACTIVE & COMPONENT
XML DATA TYPE
PERSONAL DIGITAL ASSISTANT
84
IWN XML COMPONENT
XML DATA TYPE
WEB TV
85
IWN ENGINE
86
OLAP MODULE
87
LOGISTICS MODULE
88
CLIENT BILLING SYSTEM
89
LOYALTY MODULES
90
BROKERAGE PORTAL MODULE
91
BANK/ CLEARING HOUSE
92
OTHER IWN ENGINES
OTHER IWN ENGINES
OTHER IWN ENGINES
93
B2B MARKET PLACES
B2C MARKET PLACES
95

FIG. 4

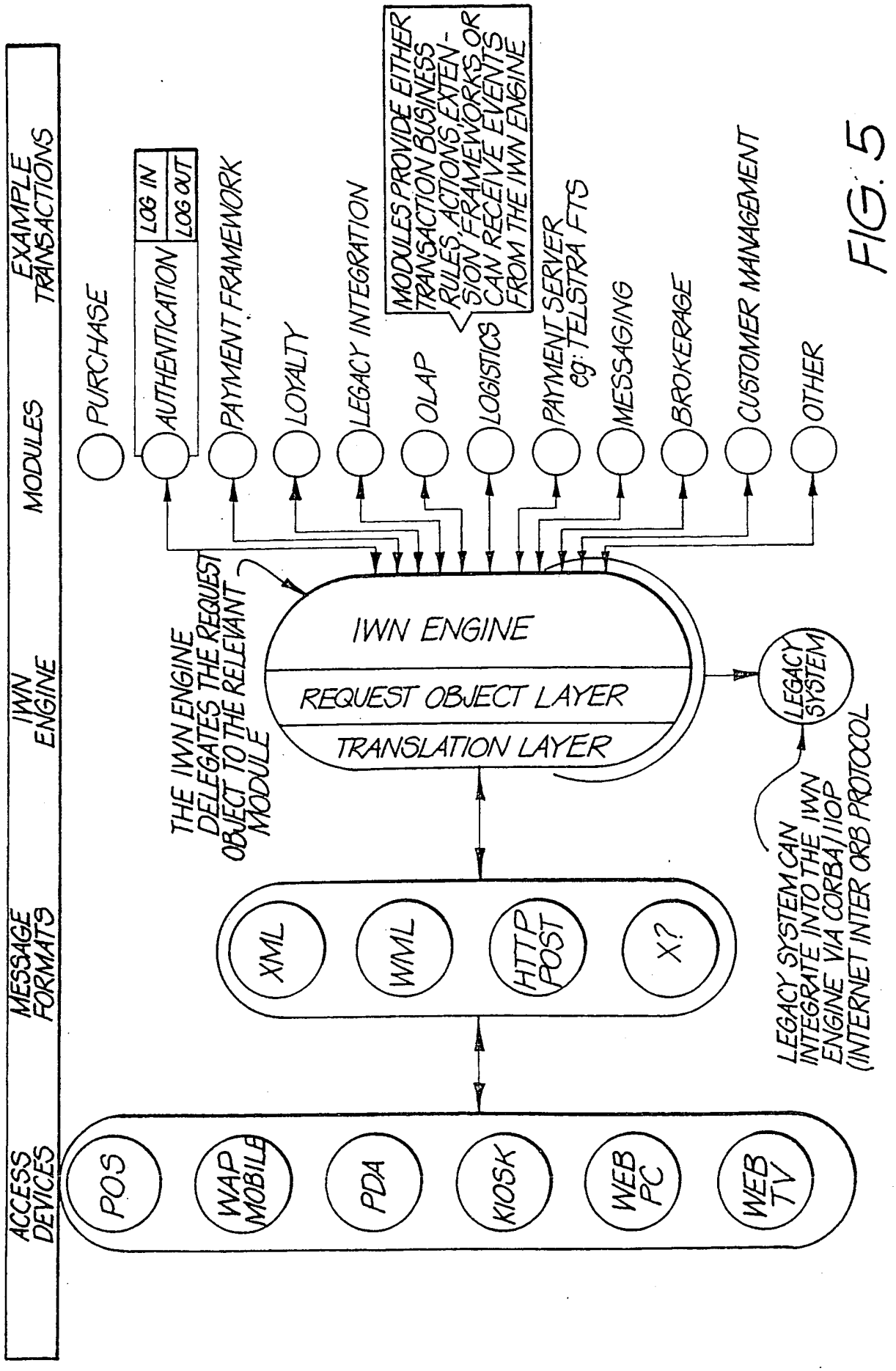
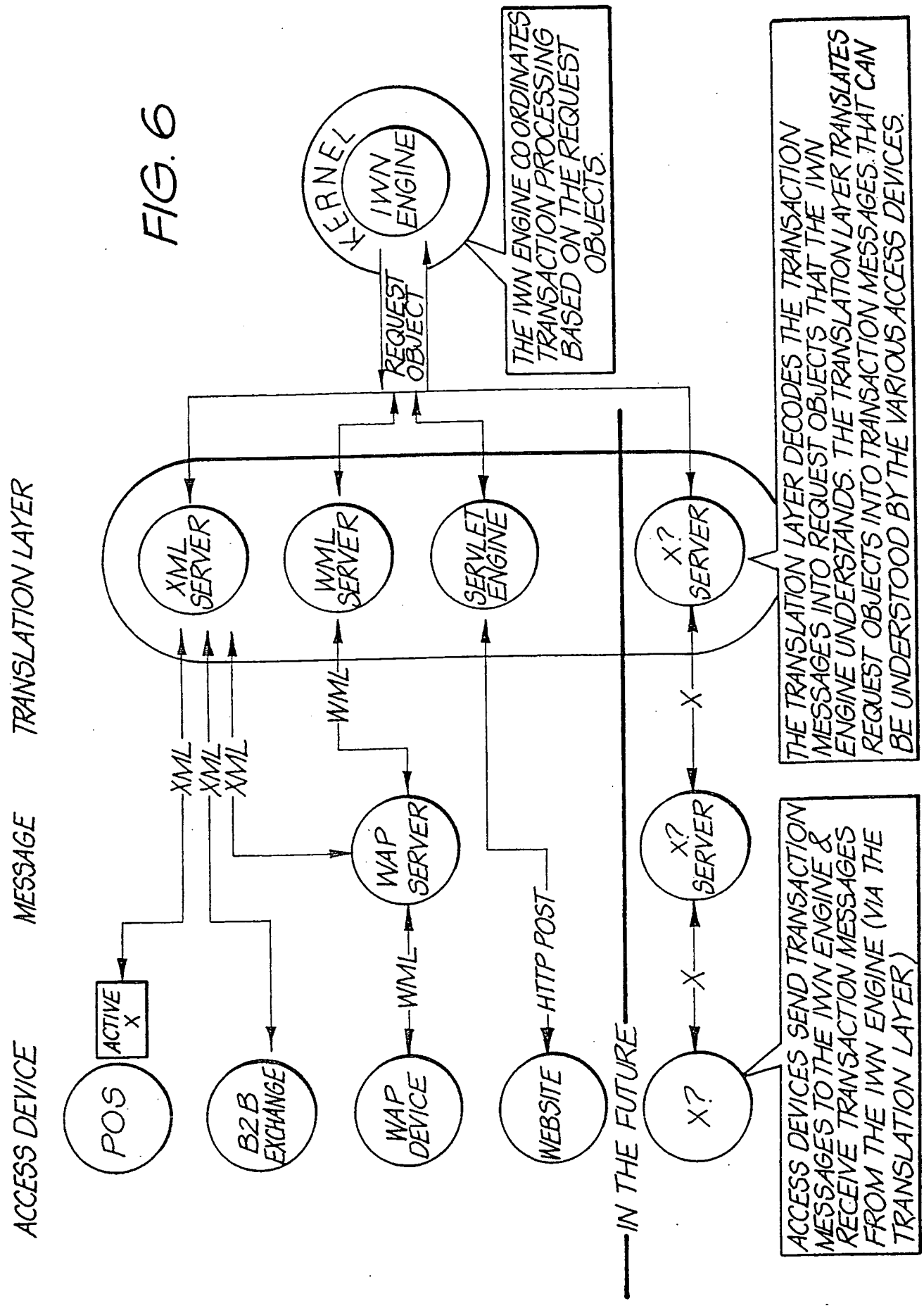

ACCESS DEVICES
MESSAGE FORMATS
IWN ENGINE
MODULES
EXAMPLE TRANSACTIONS
POS
WAP MOBILE
PDA
KIOSK
WEB PC
WEB TV
XML
WML
HTTP POST
X?
IWN ENGINE
REQUEST OBJECT LAYER
TRANSLATION LAYER
THE IWN ENGINE DELEGATES THE REQUEST OBJECT TO THE RELEVANT MODULE
LEGACY SYSTEM
LEGACY SYSTEM CAN INTEGRATE INTO THE IWN ENGINE VIA CORBA/IIOP (INTERNET INTER ORB PROTOCOL
PURCHASE
AUTHENTICATION
LOG IN
LOG OUT
PAYMENT FRAMEWORK
LOYALTY
LEGACY INTEGRATION
OLAP
LOGISTICS
MODULES PROVIDE EITHER TRANSACTION BUSINESS RULES, ACTIONS, EXTENSION FRAMEWORKS OR CAN RECEIVE EVENTS FROM THE IWN ENGINE
PAYMENT SERVER eg: TELSTRA FTS
MESSAGING
BROKERAGE
CUSTOMER MANAGEMENT
OTHER

FIG. 5

FIG. 6


ACCESS DEVICE
MESSAGE
TRANSLATION LAYER
KERNEL
IWN ENGINE
REQUEST OBJECT
THE IWN ENGINE CO ORDINATES TRANSACTION PROCESSING BASED ON THE REQUEST OBJECTS.
XML SERVER
WML SERVER
SERVLET ENGINE
X? SERVER
THE TRANSLATION LAYER DECODES THE TRANSACTION MESSAGES INTO REQUEST OBJECTS THAT THE IWN ENGINE UNDERSTANDS. THE TRANSLATION LAYER TRANSLATES REQUEST OBJECTS INTO TRANSACTION MESSAGES THAT CAN BE UNDERSTOOD BY THE VARIOUS ACCESS DEVICES.
XML
XML
XML
WML
X
ACTIVE X
POS
B2B EXCHANGE
WAP SERVER
WML
WAP DEVICE
HTTP POST
WEBSITE
X?
X
X?
IN THE FUTURE
ACCESS DEVICES SEND TRANSACTION MESSAGES TO THE IWN ENGINE & RECEIVE TRANSACTION MESSAGES FROM THE IWN ENGINE (VIA THE TRANSLATION LAYER)

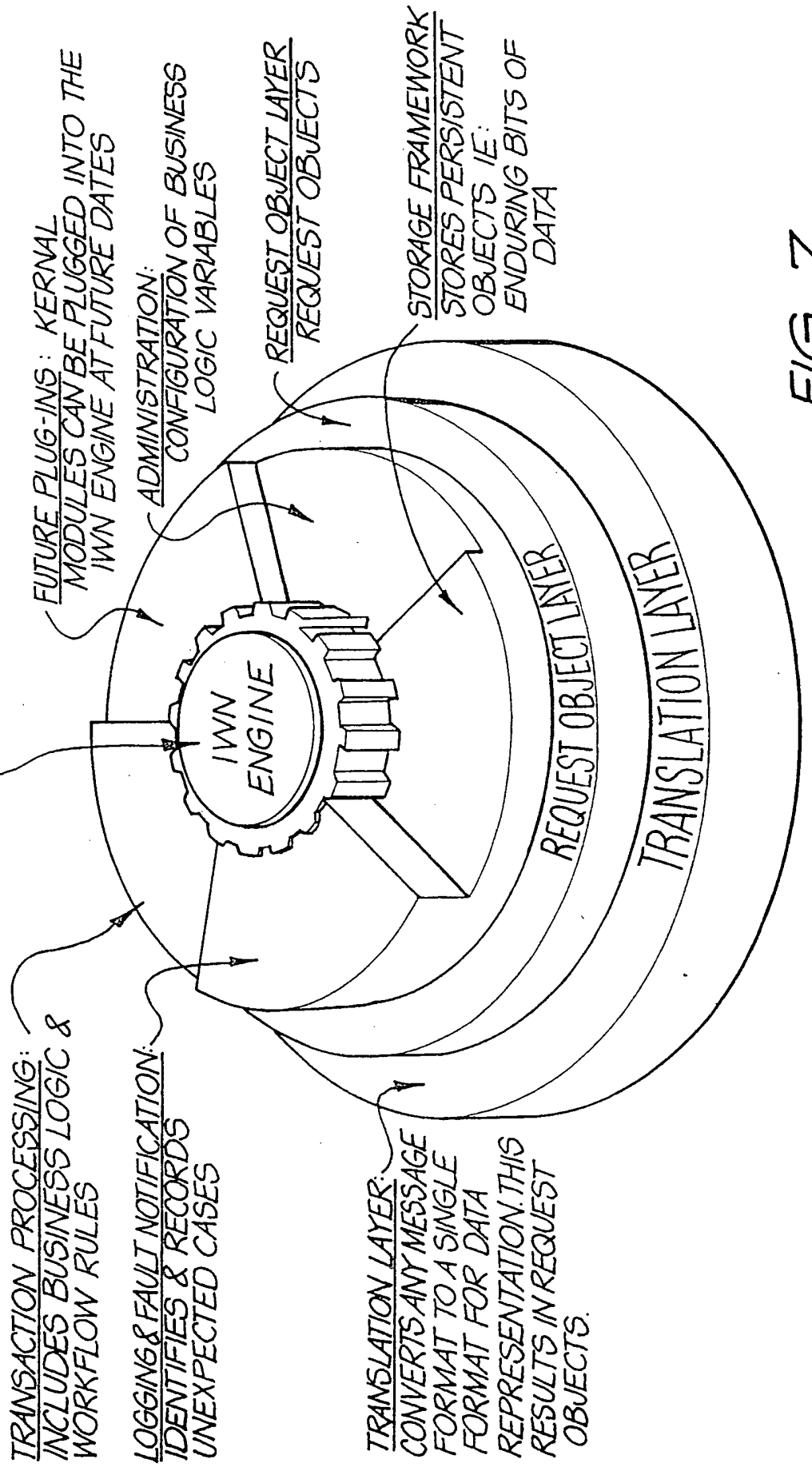

IWN ENGINE: COORDINATION OF KERNAL MODULES & REQUEST OBJECTS
FUTURE PLUG-INS: KERNAL MODULES CAN BE PLUGGED INTO THE IWN ENGINE AT FUTURE DATES
ADMINISTRATION: CONFIGURATION OF BUSINESS LOGIC VARIABLES
REQUEST OBJECT LAYER REQUEST OBJECTS
STORAGE FRAMEWORK STORES PERSISTENT OBJECTS IE: ENDURING BITS OF DATA
IWN ENGINE
REQUEST OBJECT LAYER
TRANSLATION LAYER
TRANSACTION PROCESSING: INCLUDES BUSINESS LOGIC & WORKFLOW RULES
LOGGING & FAULT NOTIFICATION: IDENTIFIES & RECORDS UNEXPECTED CASES
TRANSLATION LAYER: CONVERTS ANY MESSAGE FORMAT TO A SINGLE FORMAT FOR DATA REPRESENTATION. THIS RESULTS IN REQUEST OBJECTS.

FIG. 7

INTERNATIONAL SEARCH REPORT

International application No.
PCT/AU00/00730

A. CLASSIFICATION OF SUBJECT MATTER

Int. Cl. [7]: G06F 17/60

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC: AS ABOVE

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
WPAT (commerce, EFTPOS, database)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WO 99/08218 (Trivnet Ltd.) 18 February 1999 Whole document	1-5
X	WO 95/31789 (Visa International Service Association) 23 November 1995 Abstract,figures	1-5,11,17
Y	WO 99/28830 (Korman) 10 June 1999 Whole document	1-5

[X] Further documents are listed in the continuation of Box C [X] See patent family annex

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search
19 October 2000

Date of mailing of the international search report
25 OCT 2000

Name and mailing address of the ISA/AU

AUSTRALIAN PATENT OFFICE
PO BOX 200, WODEN ACT 2606, AUSTRALIA
E-mail address: pct@ipaustralia.gov.au
Facsimile No. (02) 6285 3929

Authorized officer

DALE E. SIVER
Telephone No : (02) 6283 2196

INTERNATIONAL SEARCH REPORT

International application No.
PCT/AU00/00730

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	WO 97/49072 (Verifone Inc.) 24 December 1997. Whole document	1-6,8,9,13-16
Y	WO 97/09816 (Wireless Transactions Corp.) 13 March 1997 Abstract,figures	1-5,9,10

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/AU00/00730

This Annex lists the known "A" publication level patent family members relating to the patent documents cited in the above-mentioned international search report. The Australian Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

Patent Document Cited in Search Report		Patent Family Member					
WO	99/08218	AU	86442/98	EP	1031106	NO	20000563
		US	5899980				
WO	95/31789	AU	25459/95	CA	2190154	US	5500513
		US	5621201				
WO	99/28830	AU	16189/99	EP	1038233		
WO	97/49072	AU	33993/97	US	5943424		
WO	97/09816	AU	71064/96	EP	848881	JP	2000500299
		US	5852773				

END OF ANNEX